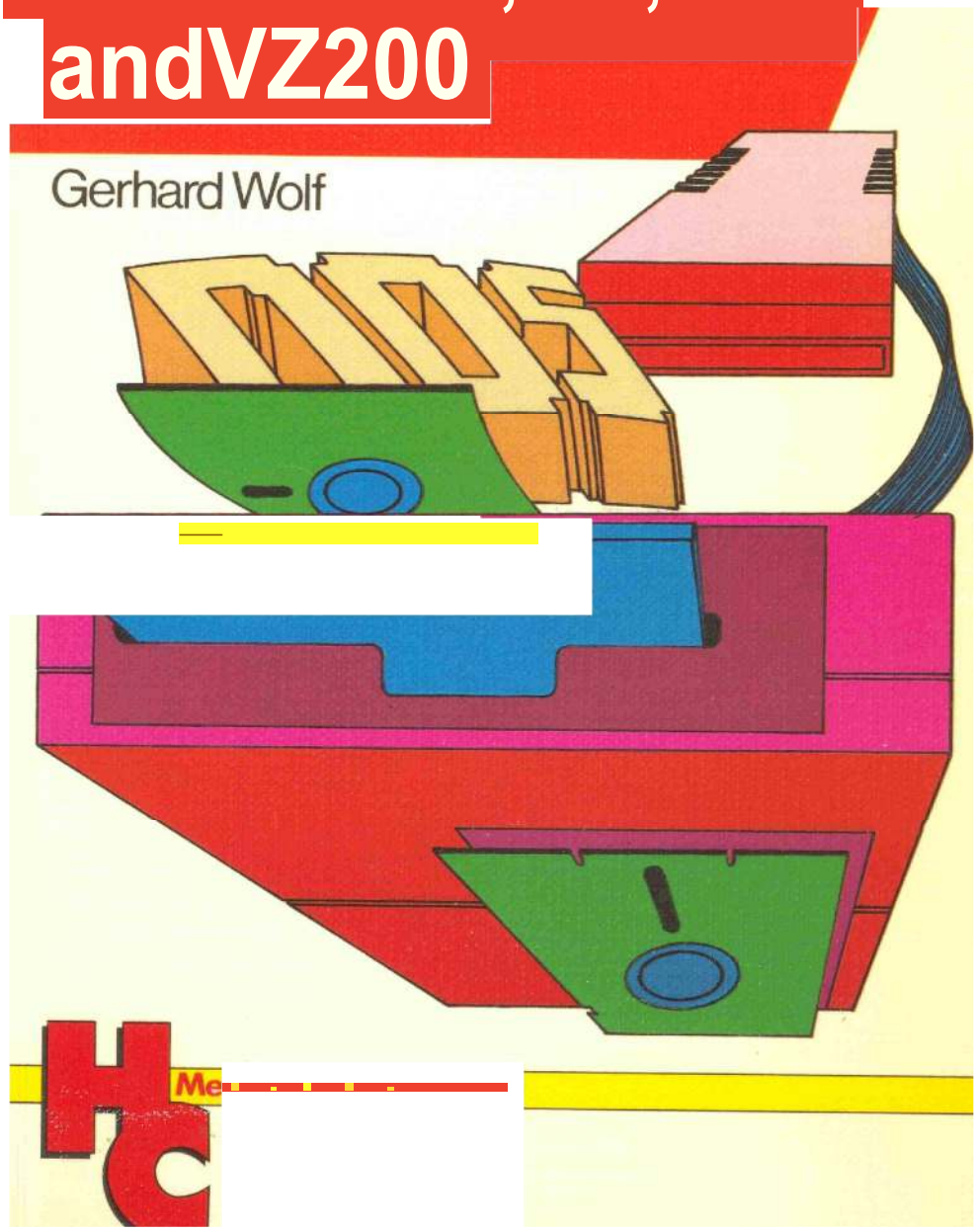
active and creative computers

The Laser-DOS for lasers 110,210, 310 andVZ200

Gerhard Wolf



Me

Gerhard Wolf
The Laser-DOS for Laser 110, 210, 310 and VZ 200

HC - My Horne Computer

Gerhard Wolf

The Laser-DOS for Laser 110, 210, 310 and CZ 200

Floppy OS Setup and Application

9

VOGEL-BUCHVERLAG
WÜRZBURG

From the same author are published:

ROM Listings for Laser 110, 210, 310 and VZ 200 (HC - My Horne Computer)
ISBN 3-8023-0852-2

The BASIC interpreter in the laser 110, 210, 310 and VZ 200 (HC - My Horne Computer)
ISBN 3-8023-0874-3

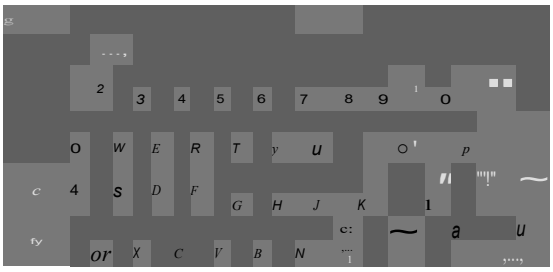
CIP short title recording of the Deutsche Bibliothek

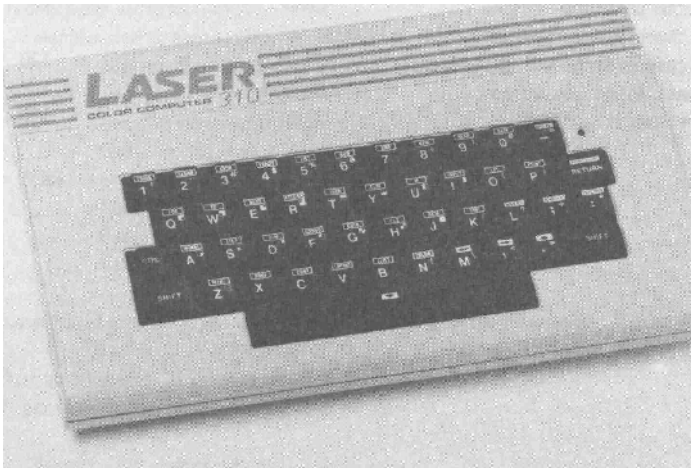
Wolf, Gerhard:
The Laser-DOS for Laser 110, 210, 310 and VZ 200: Structure u. Application d. Floppy Operating System / Gerhard Wolf. - 1st edition - Würzburg: Bird, 1985.
(HC - My Horne Computer) ISBN 3-8023-0868-9

ISBN 3-8023-0868-9 1.
Edition. 1985

Printed in Germany

Envelope design: Bernd Schröder, Böhl
Manufacturing: Alois Erdl KG, Trostberg




LASER
COLOR COMPUTER 310

h one thing to *understand* and to see through+ are solid basic knowledge1sse inaccessible, Only then is one **able** to *recognise* dependencies and *use* properly.

This principle also applies - or even more so - to the use of a floppy disk system on **a** home computer, **which** can be assumed to *be* procured primarily for commercial Z111ecke, but primarily to indulge a hobby or to train in the field of data processing.

This book specifically addresses the connection of a floppy disk system to a LASER 110, 210, 310 or V7200.

It describes in detail the basics and the structure of the system, and deals in detail with the possibilities of the included disk operating system (Disk Operating System = DOS).

It will end up for all floppy users, whether computer newcomer or savvy 'freak'.

In the first chapters the basics are presented and it 1111rd describing the usage within the framework of the available BASIC language. Then follow the detailed description (partly bit) of the data organisation on the D1 chain and a detailed description of the disk possibilities for assembler and machine programme - Kennen.

Do not be deterred by this, however, if you are faced with 11with floppy disks for the first time. The construction of this book allows the schri th, eisen entrance into the matter.

After studying *the* basics, start using the general DOSA instructions and programme/file management by using the disk only as the storage medium for your programmes.

If you are sure about this, try storing your own data from llASIC-Progra1111111en on *the* disk and then processing **it again**.

The last two chapters should only *be* signed if you are fully familiar with the DOSA application i11 BASIC and have some basic knowledge in Z8 assembler and machine language programming.

Coaponents

The floppy drive (Floppy Disk Drive) LASER DD21 ur de spezeii tur developed the use on one of the LASER computers 110, 2IQ and 310; it is also for the VZ200. These floppy drives are not compatible with other computer systems (e.g. the LASER 2001 or LASER 30 o.a.). Conversely, other floppy drives are also making tur suitable for these computers.

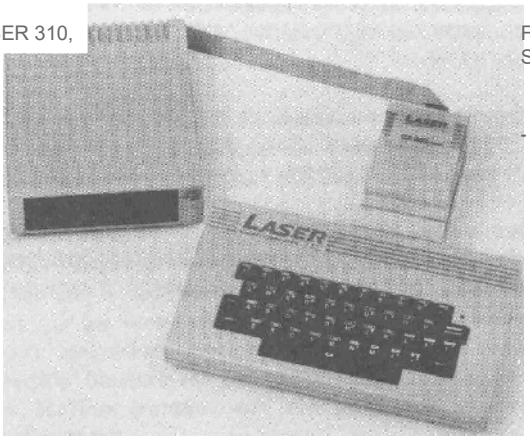
The connection to the computers is established using a floppy control (Fioppy Disic Controller) LASER 0141, which can be connected to the system bus of the Red,ner,lo: ・er, will be inserted and on the "huckepack" a memory stick {il; 1.1).

h・Drives can be connected simultaneously to a floppy drive,g,and run Jc,ssen and. The floppy control also includes the reforcier 11 ct,en S~stemprogramm extensions, the floppy operating system !Disk Operating System= DOS).

For the LASER 110 computer and VZ200 with internal 4K-RAM 1st to:.atzl:ct, e;.ne memory expansion of at least 16K required licr,.

to

Figure 1.1 LASER 310, Floppy Control, !6K-Storage1tei"Un9 and Lc>Drive



The Drive

S g1bt mne large number of different floppy systems of different manufacturers, Erne of the main i'.lass1 tiz1 faults of cheser drives is the size of the discs to be used, these vary from 3 1/2 inches over 5 114 inches to B inches. The floppy disk drive~ LASER DD:20 is a 5 1/4 inch drive that from the appearance, and also the techn1scr,en setup very much breathes the TEAC volume,

Floppy drives work with a round, rotating disk (called a disk), which, like a tape, is covered with a magnetisable bay, in which the data is inscribed by a write/read head or read out nyeder.

To be able to access any part of this disk, the write/read head must be movable. For this purpose, it is mounted on a rail, and it is visible to move across to the disc,

To write or read data 111the llopt simply moves the desired distance back, inside or upn and then grades until the desired data gets through, the rotation underneath (Figure 1.2).

The head on the slide has fixed grid positions for the overview llct, tness and readability t halber, which in turn results concentric data circuits on the disk. [The different floppy systems have between 35 and 82 rasters, the LASER DD20 has 40.

The resulting circles of 1111t data on the floppy disk are called tracks (in English ・Tracks' *i,* each track can be reached exactly immediately by the head adjustment mechanism,t 111t.

Within a SPUr 111, the data is recorded bit by bit in a row, that is, in the EDP language 'sequential'. Thus, it takes natural, nact, positioning of the head over the SPUr, still an average half rotation of the screen, b1 s the desired data erre1 cht are.

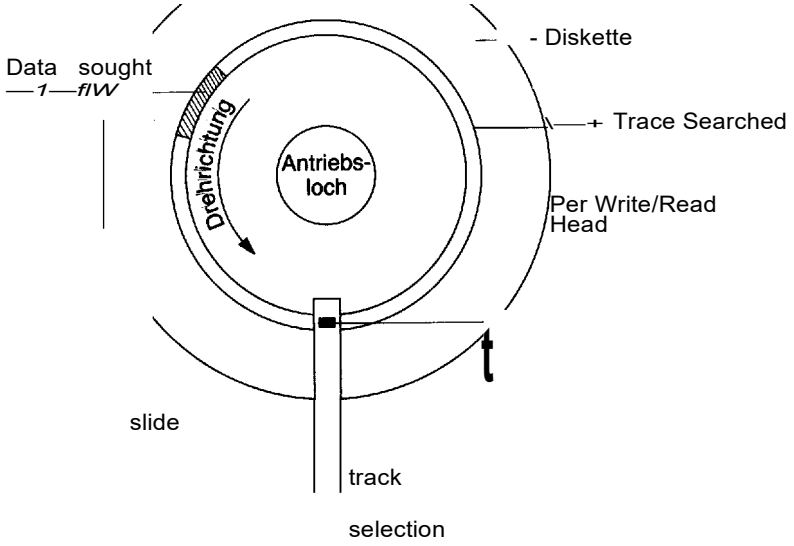


Figure 1.2 Data Access on a Diskette

So the time it takes to access the data you want depends on how fast *the* head can be turned on the particular track and how quickly the disk rotates.

Bc1111 LASER DD20 is the floppy disk 1it 80 revolutions per minute driven. The time to move the head from one track to the next is about 20 milliseconds. This results in an average access time of 51110 milliseconds.

During a write/read process 111.10 the disk should be pressed firmly to the write/write head **as** in a tape. This **is** done **with a** felt towel which, via a lever, pushes the disk from above on the head. This lever is connected **to the** $lever lock on the front of the drive. If the disk is in the vertical position (too), the disk to the head gepreOti in the horizontal position the disk is free (Bi1d 1.3).
An electronic head loader procedure, never common on many other drives, does not exist here.

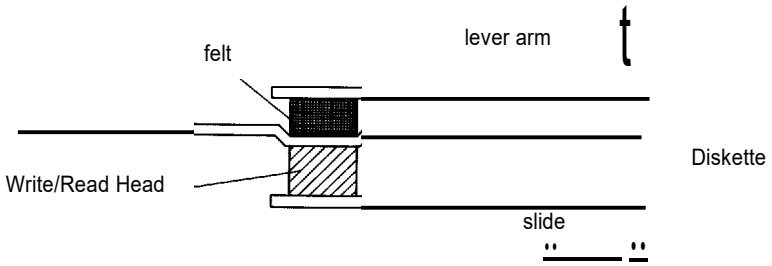


Figure 1.3 Diskette in the drive

If you've been paying attention now, you should have noticed that$ a floppy disk is always described from below, so different from what you think to push in.

The ntr 1ebsmechan1 smus is also connected to the lock}lever. When the lever (vertical) 111is closed, *the* drive hole *of the* disk is centred on a cone driven by the engine over a belt. Exactly how a disk is centred on the cone is one of the critical components of the drive.

The location of a track refers to the centre *of the* disk 1mm. Therefore, the secure writing and retrieval of the data is very much dependent on how exactly the disk is centralised. Unfortunately, the **LASER DD2I** drive has a tendency to not press the floppies exactly on the cone · Please note the help and control options mentioned in the section 'Inserting a Diskette.

The drive motor will only protect **the** floppy disks immediately before the write or write. Turned on read operations and then immediately switched off. This process will provide you with an optical indicator in the form of a light emitting diode (LED) on the front of the drive. If this is lit, you should not remove or insert a floppy disk (see Press Release).

Structure

Due to the pressing of the disk to the write/read head, it is evident *that$* no *solid* disk has been used as a basic material, a flexible, formable plastic material is used (mostly *a* tylar foil)+ which is coated with *an* iron oxide. The English name [1]floppy disk' = flexible disc also derives from this.

For better handling and protection of the coating, the disc is packed in a solid shell with a felt-like lining.
The lining fulfils three 111 essential tasks:

- Lowering the friction between the disk and the case
- Derive static charges that result from rotation
- absorption of particles of dust and dirt to protect the coating

The floppy always remains in this case, which has three openings for handling:

- a large hole in **the** centre door to the drive mechanism
- Pin oval neckline, through which the head on the coating
- access
- **a** small opening as an index hole, which is used for many drives to physika lische trace start identifier (not LASER DD20).

Al edge .the case is still a small rectangular recess, this is a write protection device that allows you to protect the contents of a disk. If this recess 11it of one of the adhesive strips accompanying the floppy packs is glued over, the writing on this disk is blocked (Figure 1.4).

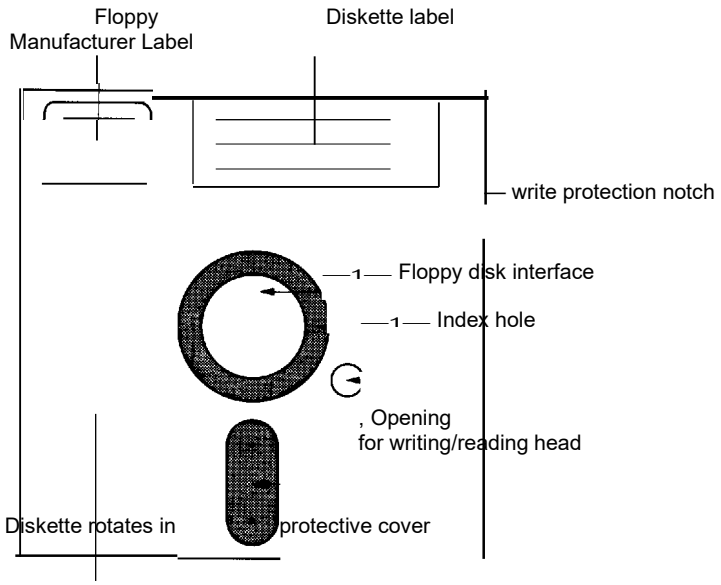


Figure 1.4 Diskette

The actual disc consists only of a very thin coating of iron oxides and is specially treated to keep u11 in contact with the writing/reading head for a reasonably long time. Nevertheless, this layer is sanded with **constant** contact with the head. A duration of 50 - **60** hours can serve as a reference. This results in a quite long life, since the Yes disk only runs on read/write access letters and the head does not always lie on the same track.

Due to this abrasion, evil tongues also refer to floppy processing as **'chip-removing data technology'.**

Insert the floppy disk

To insert a floppy disk, the $lever lock on the drive opening is placed horizontally. The floppy disk is now 11it the oval head cut forward and usually the label is pushed up into the drive to stop.

By turning the cover part in the vertical position, the drive hole of the diskette is pressed centred on the drive's front1ebskonus and pressed d1e disk to the read/write point.

Normal1se is outgoing and can start writing/reading.

However, as mentioned before, it can come to difficulties if the disk is bent on the cone 1,nrd, and this happens at the **LASER 0021** unfortunately haut1g.

The following should therefore be made a rule:

- Only lose floppy r,1 t one, reinforcing ring around the drive hole. This prevents pretzy1-like wear on the drive hole by tilting the cone.

- Centre the 1skette in the Oer case as well as 1e possible before inserting it with your hand.

- Before S1e write anything to an initialised floppy, test it with the **DIR** - command.

- After reinitialisation, remove the floppy disk, insert it, and test with the DIR command.

If, however, the drive illlfler tries to reposition the head again during a reading process, you can hear a louder crunch noise, so if the engine is running, briefly open the locking lever and close it immediately. As a rule, the floppy will move clean to the cone when the engine is running.

A disk has two sides

What floppies do you now procure for the **LASER DD28?**

As mentioned above, Ulli is a 5 1/4 inch drive, so you also need 5 1/4 inch dll chains. But there are also many different varieties again, hardsectored or graded, single-sided or double-sided, with single density or double density.

Ulll bring it to a denominator: You need 5 1/4 inch floppy disks, stacked, single hg, 111i t single density. The latter shall be labelled **"SS5";** This stands for 'single sded, single density.

Diskettes of double roof {SSDD) can also be used, these are a little more thoroughly tested, but a little more expensive for the recording method chosen by the **LASER DD2** not necessarily necessary.

If you now look closely at such a 'one-first' disk, you will notice that both sides are still coated and the oval opening for the head is present on both sides of the case.
This means that you can basically describe both sides of the floppy.

The **LASER D28** 1st drive only has a read/write head, so you have to rotate the floppy to move it to the other side. If you do this and try to write eh,as on this page, you will receive the message "?DISK WRITE PR(ITECTED'. This is due to the missing write protection notch on the other side of the case. Remember that a floppy disk is read-only when you stick the write-protect notch.
'No write protection notch' has the same effect, of course.

To use the second page only a second Schre1bschtiutzkerbe is required, which you can easily attach to the case with a hole. Remove a floppy disk other than a stencil. Don't be afraid that you will damage the floppy itself, it will not reach 111e1t into the corners of the case (Figure 1.5).

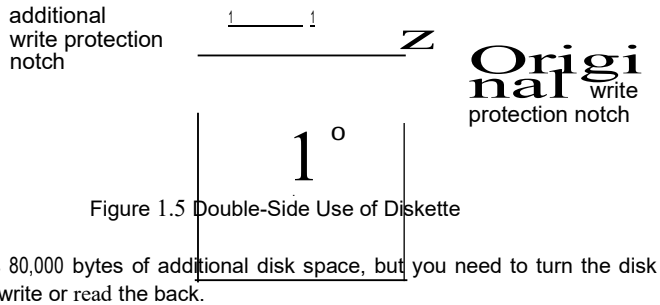


Figure 1.5 Double-Side Use of Diskette

This creates 80,000 bytes of additional disk space, but you need to turn the disk around if you want to write or read the back.

Treatment of diskettes

If you want to keep your data on the floppies as good as possible from destruction, you should always observe the following rules:

- Always keep the floppy disks in their protective covers when they are out of the drive.

- Make sure there is no 1m drive chain when power is turned off or off.

- Never bring your floppies close to strong magnetic fields (transformers, motors, magnets, TV/Mum gate, Rad1s must.); d1 magnetic fields emitted there could destroy the data content.

- Grasp the floppy only on the case. Avoid 5,e any contact with the 111agnetis1erbare coating, exposure. Also, do not try the 2u coating, scratches are fast in the surface, and you can then forget the disk.

- Expose a floppy to direct sunlight or large heat.

- Avoid contamination *of the* coating with cigarette ash, dust or other items.

- Use only one pencil if you want to label the cover. Ball pens or pencils could violate the coating through the shell.

- If possible, store floppy disks vertically on the vinyl records) to avoid pressure on the discs.

Floppy label tips.
Each disk has a label attached to it on its cover. You should use this ır for important information that does not change during the life of a floppy disk. For example, it is very helpful to add a Nu1111Der for archiving the floppies. This would have its best place there. Other useful data include your name, without the date of first use of this disk,

For content information, use the Each Diskette Pack with adhesive labels lying down, which you can easily replace. If you do not stick any 1 heavenly kills with it, you have the whole shell surface,e at your disposal.

recording structure

How a disk determines the amount of data you can store, h1e' Each system has its own storage capacity for one disk. 5 1/4-inch floppy disks can have up to 1/2 million bytes (characters) per floppy disk. At 111 LASER DD21 it is a bit more than BOO bytes.

Two crucial factors influence the storage capacity, This is once the number of rasters with which the head moves over the disk and the same as the number of data tracks to be described on the disk 1st. Variations between 35 and 80 on different systems are currently known.
The LASER D28 has 40 tracks.
The second factor is the way the bits are written to the disk. Here one distinguishes between "simple density" (FM) and "double density {MFM), double writing density also equals about twice the capacity. Leg LASER DD21 is recorded with simple density as mentioned above.

But the storage capacity could still be nearly twice as large if the data were written to the disk in the same way as it is stored without 111 additional measures in a row. So you can get a lot of data on the D1 chain, but you can't do much with it anymore. You wanted to find out in a sausage of bits a very high-quality information without having to go through it all the time.

The advantages of floppy storage can only be used when the OnDrawings are meaningfully structured by dividing them into small manageable units, of which one 111white, 1110 they are on the disk. This is the only way to take advantage of the un1indirect access capability. This means nothing more than allowing the records to be formatted.

Such a formatting is achieved by dividing the record on the disk within the 4 different tracks again, 111ie at a cake, into 16 equal sections (sectors). Each of these sectors is separately addressable and can be treated individually who the.

Each track consists of 16 sectors, in which 128 data bytes can be accommodated (Figure 1.6). This means that the **LASER DD28** has an exact storage capacity of

40 tracks x 16 sectors x 128 bytes = 81928 bytes

per diskette side.

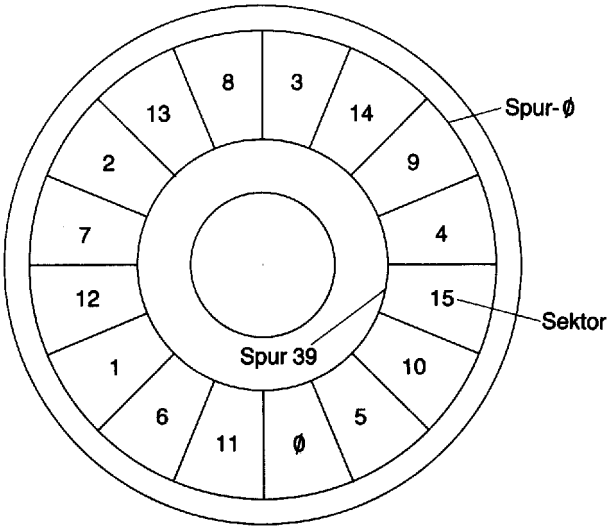


Figure 1,6 Location of tracks and sectors on a disk

This is not all, however, this must be stored on a formatted floppy disk.

In other words, to be able to access a sector directly without gray stands, one must "know when the desired intonation just goes by under the head.

For this purpose, each sector receives a header, a so-called AdrePfeld, in which the sector nuns and, and head adjustment errors can be detected, also the Spurnuner.

To detect recording errors in a sector one more test sunfield is added at the end of the sector.

But this alone is not enough. Only rarely does the head stand there, 1110 a new byte on the track begins. In general, it will start reading in the middle of 111 bytes. However, since the data are stored in a row, no matter how much it is, it is impossible to identify the beginning of a byte.
This means that the first time a record was found, tan refers to a synchronisation of the head.

Specifically defined BiStrings and AutoDrawSR1 markers are written to the floppy disk, which have an easy to recognise pattern.

There are z111e1 different types of these brands. One is in front of Every sector address field, that is the address mark'; a second is in front of Every data field of a sector, the 'data mark'.

Each of these brands is preceded by synchronisation bytes, and the brands are immediately followed by the data. This makes it possible to distinguish clearly whether s1ch is located in front of a data record or an address field.

More space is lost on *the* disk due to 'recording gaps' that are located behind each data field of a sector. These gaps are urgently needed to compensate for fluctuations in the rotation speed within certain limits (Figure 1,7).

Such a basic structure of the disk must be created before any data descriptions. This process is called 'initialisation; a separate command is available. When initialised, the subdivision is made into sectors and all address and data markers are stored1.

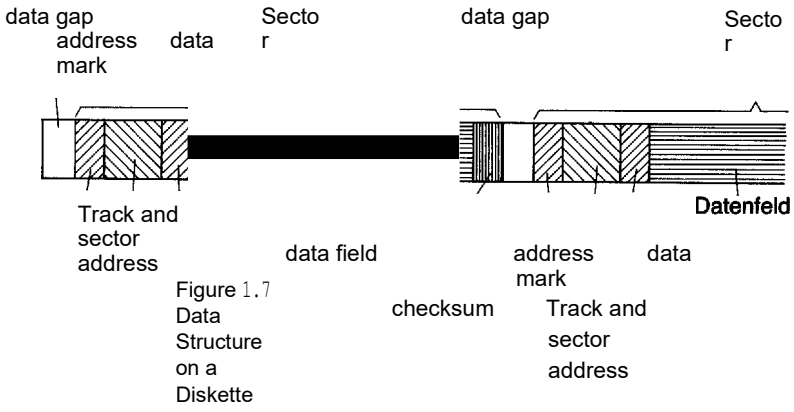


Figure 1.7 Data Structure on a Diskette

Figure *1.b* shows that the sectors are not numbered continuously by 1 b1s 1 but rn 3 jumps on the disk. Through this little trick it is mogl1ch, while lill!fil: Reversing the disk several consecutive sectors and thus speed up the Train1 ft.

After these explanations, it should be understandable how the computer can find each sector on the disk.
But you usually want to know garnicr,ts from individual sectors, you are looking for e1n 93nz specific programme on the floppy or file you have created there. On a floppy disk, you will usually have more than one programme or Ine Date1 saved. How do you get to exne such complete notice, Dhne himself to have to drive book on sectors?

A whole trace of the chain was sacrificed. On the track, the outermost track, is created a lnhal directory of the disk, in which is recorded which programmes and files stored on the disk and 1110 to find them srno. With the DOS command 'DIR' you can output this table of contents to the B]dsh1r.

The last sector of this track @still has a special use in it w1rd noted for each sector of the disk, whether it is free or with valid data included 1st,

The Floppy Control

To connect a drive to a computer of 1111rd, a floppy disk controller of type **LASER D148** benot1gt.

[1se Ird connected to the system bus of the computer and has at the back side ze1 2 pin1ge connector for connecting one of the ze1 drives.

The task of the floppy control is to implement logical orders quickly, 1e e e.g. 'Read a programme' in individual steps for specific control of the sound111erke, e.g. step pulses for track adjustment, read a bit, write a bit, engine on, engine off, etc. ·

The floppy control also includes the floppy disk operating system!I! (Disk (tperating S! jstem), called **DOS** for short. This is stored in BK-ROMs and adds 17 commands to the BASIC language range of the computer, which are required to operate the floppy disk station. These include "INIT" for floppy initialisation, **"SAVE** and **'LOAD'** for saving bz111. Load a BASIC programme.

This floppy drive system is addressed via d1e Addresses 400 5FFF (hexadecimal), which have been retained for this extension purpose. A 31 byte is reserved except for the one at the end of the RAM -

Installing the Floppy System

In principle, all connections of electronic components should be connected only when the power supply is switched off. How fast 1 is otherwise very expensive high-integrated brick through, occurring voltage peaks destroyed.

This also applies to the connector of the floppy system. So make sure that d1e power to the computer system is off.

The 1111rd floppy drive was first connected to *the* system bus on the back of the computer. The 1st d1e plug-in strip, *which* was previously connected to a possibly existing memory card of the device, can now be connected to the top of the floppy control.

The back of the floppy drive contains z111e1 2111-pin connectors for connecting *the* drives. You are labelled Dl · for drive 1 and D? for drive 2.

If you have S1 only in drive+, connect it to 'Dl'.

Each drive requires a disconnected power supply. Plug the 5-pin DIN power supply 1st connector into *the* back of the drive.

Now connect the power supplies to a power outlet, so your setup is complete.

You should make sure that no floppy disk is in the drive when connecting or unplugging the network connection, as it could destroy data content 111.

If you do not work with the system for a longer period of time, you should stop the power to your floppy drives.

It is recommended to shade all power connections of the Recr,ners~stems a 220 volt plug strip with illuminated on-i switch, so can 5e te:
Absct,work must be cut by simple push of the entire power interruptions.

Remember that the LASER I 10 and VZ20 computers with internal 4f RAN must have at least one I6K memory expansion,

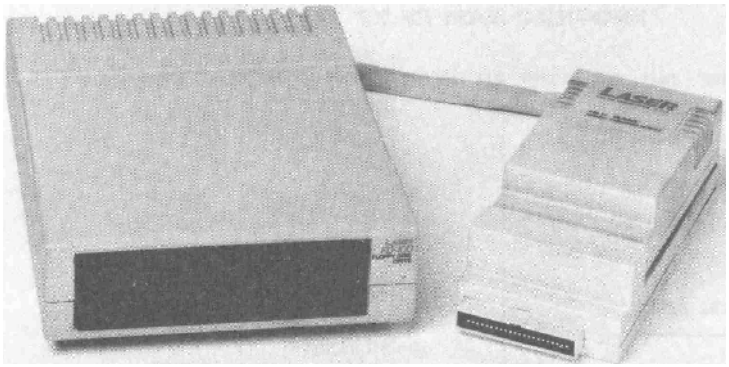


Figure 1.8 Floppy Drive, 16K Memory Expansion and Floppy Control

Systea lnitisation

After successful installation, turn your computer on like ge!LIDhnt ern.

The initialisation routine of the computer erf:automatically detects that a Dlsi:ettenSystem is connected and briefly starts the engine of the drive connected to Dl. The write/read head is positioned on 1e track {these are the clack noisesJ.

The text 'BASIC V2.8 or 'BASIC Vt.2'

1rd through 'DOS BASIC Vl.l' is replaced.

This is for S1e the safe toe that your floppy operating system is mm1t1al1s1rt.

It is now recommended that you use the above mentioned additional 17 beds for D1sketten-ilearbe1.

This does not happen, of course, 111If you have not connected an additional 1memory111 with a LASER 110 or VZ200.
Instead, the message appears
'?INSIFFICIENT IENORY FOR DOS'.

You can work with your computer as you are, but you have i:a possibility to address the disk.

Technical Data

diskettes	Standard 5 1/4 in S5SD (s1ngle sided, smngle dens1tg)
recording format	serious single-point D1 real 40 tracks 16 sectors/ trace 128 bytes / senor
Capacity	80Kb
Drive	80 revolutions/minute
Str0111Supply	+ SV, +12V DC voltage over. separate power supply

Building the LASER-DOS

The LASER floppy operating system, or DOS (Disk Operating System) for short, is an additional programme package of approx. 8 KByte+ that has been placed in the 1n R module in the disk control housing,

It occupies the memory range from address 13B4 (40MH) to address 24575 (5FFFH), which was retained in the LASER computers for such extension purposes (Figure 2.1)

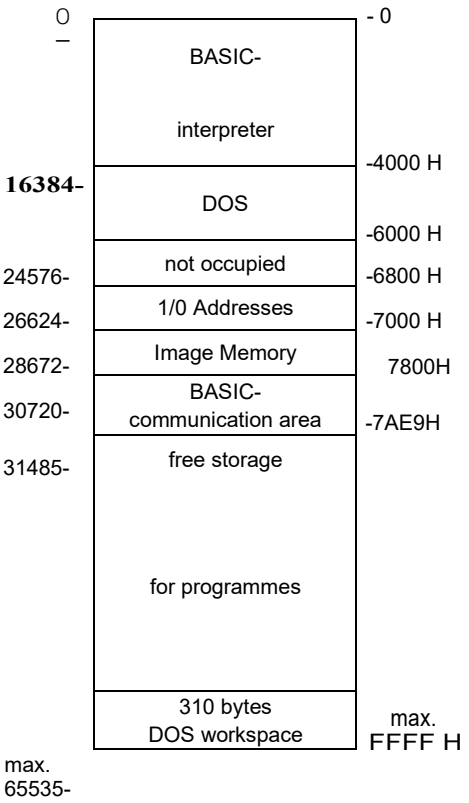


Figure 2.1 The LASER 110, 210, 310 and VZ280 memory allocation

When the computer is turned on, the presence of the Betr1bs5systems auto111system is detected and initialised by the default ROM's initialisation routine, i.e. embedded in the BASIC interpreter's flow routines.

The floppy operating system has its own comman ^o interpreter, which independently detects the 17 additional input commands and initiates its own execution routines.

The additional commands are only floppy operations that allow you to save and recover programmes and also to keep your own data on the floppy disk er@glicht.

DOS - **Kanaan dos** Support

Koando - Syntax

The available commands are entered without any special markings such as standard BASIC statements.

Most of these K0t11111andos can be used in BASIC programmes, as well as in direct mode, i.e. for immediate execution of VOit screen off.

However, a **few** are limited to one mode or another. When exactly which K011111ando can use 111erden is noted in detail in the detailed descriptions,

From the syntax, the Kot11111andos can be classified into three categories:

- COMandos that do not address a file

  Koaando C paraeter J

- COMandos that target a file

  Koaaando Filenase [**.paraneter**]

- Commandos that target two files

  kaaado "File Title", "File Name2"

Par parameters are additional information, d1e required by some commands. If several parameters · ertorder l I, then these are separated by comma.

A small limitation is found in be1 of the application within BASIC programmes.
The additional floppy commands are not detected, 111if **they** are specified directly after 1nem **THEN** or **ELSE when** using IF statements.
You must always use a stand-alone command either at the beginning of a line or after a Kornmando separator ·: · are entered.

```
100 IF A =        THEN RUN ·XYZ                    false

100 IF A = 1 THEN :RUN ·xyz·                       right
```

or

```
100 IF A"?I THEN 120 110
RUN ·XVZ
120
```

File Types and Specifications
In LASER-D, the dre1 is distinguished by different types of files:

- RAS] Programme Files
  marked "T" as a file type (=text file).

  In this file type 1J.1BASIC programmes are stored on the disk.

- machine programme data
  marked "R" as a file type (=binary file).

  In **this** file type, machine programmes are stored on the disk.

- Data Date1n
  marked "D" as a file type (=data).

  In this file type, your personal data will be stored if S1e want to put it out on the floppy disk in the au5 InM RAS]C programme.

BASIC and Mashtun programmes are stored on the floppy disk in 1m of the same format, D different type identification is only in the table of contents and beI,11rkt a different treatment during loading and starting.

Data files have a completely different structure, so there are also restrictions on the application of individual commands.

If you want to access or create a file on the floppy disk, the commands d1e specify a file name ertorderl id, which will carry into the lnhal tsververzeichnis1s of the In9disk.

A file name can be up to eight characters long and can consist of any number, number, or number sequence.

In the 1st commands, always specify the file name in quotation marks. The final quotation mark 1 must not be forgotten, even if no further information is given.

Unfortunately, the LASER-DOS does not allow the use of external string variables instead of the file name; this must always be indicated completely directly in the command. This makes it difficult for d1e to handle various data files flexibly. How you can still help yourself is noted in chapter 5 'Tips for Programming'.

The 17 additional ict,en floppy,mandos let s1ch funkt1nell 1n classify three groups.

General instructions11ngen

INIT	Upgrade a Diskette. **t1**t h1sm Commando rd de basic structure brought to the dskette, d.t, se r 1n tracks and sectors unterte1 lt.
DRIVE n	Drive Selection. Hermt you can select one of the ze1 anschi1 drives for further processing.
DCOPY	Copy floppy disks. With this command S1e copy the intait of one disk to another.
STATUS	Output floppy status. **With** STATUS" you can print the available space on the floppy disk. (only from DISK BASIC V 1.2)

File filtering capabilities

DIR	Print the table of contents. All programmes and data stored on the disk will be listed on the screens.
SAVE nane	Secure a BASIC programme. A BASIC-P,ogram containing IIII memory is written to the disk 1111t named 'name'.
LOAD • naae	Load a BASIC programme. The BASIC programme labelled IIIIt 'name' is read in by the floppy disk.

role nae •

Download and start a BASIC programme.
The BASIC-f'programme named 'name' is read from *the* disk and started immediately.

BSAIJE • naae • ,aaaa,eeee

Securing a Machine Programme
A 1m memory machine programme **nrd** with the file name "name written on the disk,

BLOAD nae

Load **a** machine programme.
The 111i t 'name' specified measles, nenprogramm is read from the floppy disk.

BRN "nae

Charging and starting of a serious machine prog• •amms.
The III t "name• specified machine programme is read and started from **the** disk.

REN 'naae1 • , 'naae2'

Rename My Data.
The Mt name will be renamed 'name2' on the floppy disk.

ERA • naae •

Delete a file.
The one with 'nall!' deleted the named file and the file or floppy disk.

DCOPY • naae •

Copying a Prograns.
The 111it "name• designated BASIC or Machine Pro1111111 is copied to another disk.

Save 1111d Processing Data

OPEN • naae • ,11

Open a data file.
Open the data file1 1rd labelled 'name' for writing or reading.

PRt • naae • ,var1Cvar2 ... ,var11}

Write to a data file.
The variables specified **in** the command are merged *to* a data set and the data file **'na.e' is** written 1n.

INI •naae•,varl[var2 • **• ,varn1**	Read from a data file. From the file 'name' a record is read and transferred to the specified variables.
LOSE "nane"	Close a data file, The data file labelled 'name' **will be** closed.

Allgt!Instructions

INIT

Preparing a Diskette

Syntax: INJT

as Direct-#ommward and 1m Pr ogr on-mode.

The INIT command prepares a disk tGr d1e storage of programmes of data; It will be initialised".

This means that d1e 1111 section "Recording Structure Represented Ite basic structure is manufactured on the disk.

40 tracks 111it per 11!e1ls 16 sectors are set up and all Aorejund data brands written.

After writing, each sector is individually addressed and checked.

The entire lni tial process n111,mt about 2-3 minutes in demand.

After correct execution 111BASIC will log in with READY, and the next l\ommando can be entered.

The Inihal1sation process9 can be cancelled at any time by pressing the .BREAK key.

Act,tung:

1t the]NIT command will overwrite an n1write-protected disk without any 111e1tere check1, i.e. maybe betJndl1che data will be lost.

ogi1che errors;

7DISi-. wR!TE P.ROTECTED	The shield 1 notch of floppy 1st overdebt.
[7]DISK 1/0 ERROR	An error occurred while reading the control. (tet, bad floppy or bad centring - see Inserting)

DRIVE

Drive - **uatl**

DRIIJE **n**
n = Run111erk (1 o. 2)

as direct command and 1M programme todus.

The DRIVE - iomando is used to select one of the two connectable drives.

After turning on the computer and after each Kop1erk011111ando IDCOPYI 1 drive 1 is automatically selected.

If you want to access drive 2, then nut$ first 111i t DRIIJE 2 will be switched on it.

All DOS—KOIM!andos, aujer DCOPY, 111are selected on delll, I ten drive out..ihrt. S1e therefore make sure that S1e halls one the correct drive selected. A [1]INIT"-K0111111ando e.g. on the wrong drive inevitably leads1ch to the destruction of a disk there random1g with 1cht1gen data,

If you are not sure which drive is currently selected 1st, please run a DRIVE-Komll!ando (DRIVE or DRIVE Z) as a precaution.

The DRIVE command. only changes the DOS internal pointers, no chain access occurs.

Possible errors

?RJNCTION CODE ERROR	incorrect drive failure 1 or Z)

DCOPY

Copy D1

Syntax: **DCOPY**
only allowed as direct K0111111ando.

The DCPY command without further parameter specification leads to a complete copy of a floppy disk to a second initialised floppy disk.

Copying is possible with one or two Laufli!erken. On a111 drive 111, however, you have to push the floppies several times during the copying process.

After entering the Koman dos, you will first be **prompted** for the Aus111atd of the source and **destination drive.**

SOORCE DISK U/21? **DISK**	(Source = Source)
DISK 11/21?	(Destination= Destination>

Answer each of these questions by pressing **'1'** or '2'.

If you only have one drive, then answer each question with **'1,**

With CTRL/BREAK the Ko111mando execution can be cancelled.

After the drive selection, the copy process begins, using **nrd** the entire RAM memory earlier, and then having to switch to the first output and drive as possible.

Copy from one drive to another, so the entire copy process runs automatically. With only one sound1J1erk (from 1 to 1 or 1 respectively) from 2 to 2), before each read or write operation, you get 6elegent;e1 t to insert the correct disk.

INSERT SOURCE DISI<ETTE
<PRESS SPACE WEN READY)

before every read from the source disk, bzlll.

INSERT DESTINATICN
DISKETTE !PRESS SPACE WEN
READY)

before each write to the target disk.

You can stop the copy at any time by pressing the BREAKT key.

The copy process is displayed with **READY**.

– Note that} the target disk must be initialised first.

– Data on the target disk overwritten 1J1erde (to rich-
actual runs (111erks and diskettes1J1ahl)

- The **entire** available RAN range 1111rd overwritten by DCOPY, i.e. data stored there or Progra11111e must be backed up beforehand or then **reloaded**.

– When using Extended BASIC", a neumn1t1al1sation of the computer is required (power off/on}.

– After completion, regardless of a previous DRIVE command, a drive 1 is selected.

possible error	
?ILLEGAL DIRECT	Attempted to invoke the DCOP'Y command from within a programme.
7DISK RITE PROTECTED	The write protection notch on the target disk is superglued.
?DISK I/O ERROR	Write or read errors on one of the two floppy disks. (defective or poor centring)

Note
This is one of the most important commands of the DOS.

As mentioned above, no disk is a reliable data store (abrasion).

So make **a** copy of every disk that contains important programmes and data for you

- after initial creation or acquisition
- after any substantial change in content.

STATUS

Floppy Status **Output**

(only from DISK BASIC V 1,2)

Syntax: **STATUS**

Approved as a direct command and in programme mode,

The STATUS KOMando detects and outputs the space remaining on the disk.

The output takes two forms. In the first place, the number of remaining free sectors in *the* form

nn RECORDS FREE

output.

In the z111eite line, the free bytes in the forw are specified,

nn.nnn K BYTES FREE

Example'

```
STATUS
80 RECORDS FREE
10.0 K BYTES FREE
```

Possible errors:

DISK I/O ERROR	Unable to read floppy disk occupancy summary.

File - IJl!Admin.II

DIR

Print the table **of** contents

Syntax: **DIR**

Approved **as direct koando** and **in programming mode.**

V DIR-Kosando will display on **the** screen a directory of all **programmes** and **files stored** on **the disk.**

The listing includes File!!P and File Names. daily

file types;

T = BASIC programme (lextfile)
B = Machine Programme D = (binary)
Data - File

Be1 game:

D:KALAH
T:AB.LAND
T:ANSCHAC
H
D:KARTEI
READY

The disk contains
- zıdıı Machinery Programmes, SCHACH and KALAH,
- two BASIC programmes, AB.LAND and ANSCHR; and
- a data file lllit named KARTEI.

The collection can be paused by pressing the Space key (SPACE) and continue ~1eder.

Possible errors:

DISK *IN* ERROR — The contents of the floppy disk could not be read manually.

SAVE

Backing up a BASIC programme to disk

SAYE • nae •

name = ma>:irnai 8-digit programme name,
in Annexe rungsze1ct, en.

Approved as D1rektko!llfflando and in programme mode.

Ern 1m Spei rather oef rndl i ches BASJ C programme 1111 rd stored under the file name 'name' on the disk.

The programme receives the type identifier "T' (text file).

In direct mode 1111rd, the completion of the save operation with READY is suspended.

In programme mode, the programme will play with the beter following 'SAVE':! .

Example'

SAVE 'KARTEI'

will transfer a 1 metre BASIC programme under the name "KARTEI' aut to the disk.

Possible errors:

SYNTM ERROR	- no data name specified - File name not quoted - No end of line after file name (RETURN) or command separator ': ・.
7DIS RITE PROTECTED	The write protection notch on the floppy disk is glued.
iflLE ALREADY EXISTS	A file with the same name already exists on the disk.
?DIRECTORY FULL	There is no more space in Table of Contents 1st (maximum 120 entries).

2DISK FULL	There are not enough nzahi trer sectors on the disk for the programme;
DISK 1/0 ERROR	There was an error writing or reading on the floppy disk.

Writing can be cancelled at any time by pressing the BREAK key. Depending on when the key is pressed, however, the listing in the table of contents is not always deleted (errors in DOS).
In order to ensure proper D1chain management, you should therefore check the table of contents with DIR in such a Fa] l and delete the file with ERA by hand if necessary.

LOAD

Load a BASIC programme from disk

Syntax: LOAD ・naae・

・name' = maximum 8-digit programme name,
enclosed in quotation marks.

Accepted as a direct command and in programme mode.

A BASIC・ programme stored on the floppy disk under the filename 'name' will be loaded into memory.

The completion of the loading process is displayed with READY,

Beispiel:

Car LOAD

the BASIC programme carries KFZ from the floppy disk in the game.

You can then view a BASIC programme loaded like this in m1t LIST and if necessary, specify 1110.

Achtung:

Before writing a 1110di tiz1 programme back to a floppy disk, you must either delete the programme m1t "ERA" that is located there beforehand or give the modified programme another name.

Example'

```
LOAD
'XYZ' >REA
DY LIST

      Modify

ERA
'XYZ' >REA
DY SAVE
'XYZ'
```

Make **the** programme read into the direct mode (BASIC-armstart), regardless of whether the call was made directly or from one!!II PrograN.

You can stop reading at any time by pressing the BREAK key.

Errors:

?SYNTAX ERROR	– No filename specified – File name not in quotation marks included – After 'name' no end of line (RETURN) or commando separator ':'.
FILE NOT FOUND	- No Progra11t111 with detailed information could be found on the disk.

?FILE TYPE MISMATCH	- On the disk a file of the same name was found, but this is not a BASIC-Progra11111t (File type= Tl.
?DISK I/0 ERROR	– There was an error reading from the floppy disk. (floppy disk failed or centring test]n

RUN

Loading and launching a BASIC programme11111s

Syntax: **RN 'naae'**

'name' = maximum 8-digit programme name;
enclosed in quotation marks.

Accepted as Direct Canadian and in Programme Togo,

A BASIC-Progra1111 stored on the floppy disk under **name** will be loaded and run into memory.

Example'

GRAPHIC RUN

The BASIC programme "GRAFIK rd loaded and executed.

Possible errors:

?SYNTAX	– No filename specified
	– File name not in quotation marks included
ERROR	– After 'name• no end of line (RETURN) or COM separator • : • .

FILE NOT FOUND	- No programme with delli specified name could be found good.
?FILE TYPE NISHATCH	– A file of the same **name** was found on the disk, but this is not a BASIC programme (File Type= T).
7DISK I/O ERROR	– There was an error reading from the floppy disk. (floppy or centralisation problem)

BSAVE

Securing a Machine Programme on a Diskette

Syntax: 1SAVE **naae aaa, eeee**

'name' = maximum 8-digit programme name enclosed in quotation marks.

aaa = Initial address, 4-digit1g; in hexadecimal writing.

eeee = 4-digit progran end address, in hexadecimal write.

As direct command and **in programme -t%us** approved.

A 111th **i**memory machine programme is written to the disk from **'aaa'** to 'eeee' on the filename.

In the table of contents it receives the type designation 'B' (binary file).

In direct mode, *the completion* of the save operation with READY **is** displayed I **Progranodus the programme is** continued **with the** command following BSAVE.

-

Instead of a machine programme, this command can also transfer any memory area to the disk and then load m1t BLAD again.

Only BRUN requires a downloadable machine programme, since this started immediately after loading 1111rd.

Resp1ei'

BSAVE 'BOWLING',8000,94FF

The machine programme 'BOWLING' is available from
Transfer 8000H to 94FFH on the floppy disk.

Possible errors:

7SYNTAX ERROR	- No filename specified - **Filename**,nct,t in quotes - Start and/or end address missing - Start or end address not 4-digit hexadecimal (@-F) - Parameters not separated by commas, nt.
7DISK RITE PROTECTED	The write protection notch of the chain is superglued.
?FILE ALREADY EXISTS	A file with the same name already exists on the disk.
?DIRECTORY	There is no more space in table of contents {max1x 12 entries). There is not a sufficient number of free sectors on the disk.
FULL? DISK	There was an error writing or reading on the floppy disk.
FULL	
7DISK I/O ERROR	

The write process can be cancelled at any one t, pressing the BREAK key will be cancelled, hang1g from the time the key is pressed will not be deleted 111,more the entry in the Ynhaltsverze1chnis !Error 1111 DOS}.
Uw,to ensure a non-stop floppy disk management, you should therefore check the lnhaltsverze1ch01s 1111t DIR in such a case and delete the file with **ERA** if necessary by hand.

BLOAD

Load a Dskette machine programme

Syntax: **LOAD • naae •**

name" = 8-digit maximum,
enclosed in quotation marks.

Approved as Direct Klell!IIRando and in programming mode.

A machine programme 'III stored under the filename 'naJRe' on the disk is loaded into memory.

A direct111ando ~displays the end of the loading process with **READV, 1** programming mode continues the programme with the aut BLOAD command.

Beisel

UPRO1 BLOAD

Maschinenprogra11111 **UPR01 is** loaded from the disk.

The KO111ffjando is primarily suitable for loading 111it BSAVE stored machine programmes from a11 BASIC programme and calling them via USR as a subroutine.

Example

```
220 BLOAD 'UPR01': 'LOAD SUBPROGRAMME 230
POKE 30862, 0: 'LSB STARTUP ADDRESS = 00
240 POKE 30863,176:        'MSR STARTING ADDRESS = RO
250 A= USR ( 0 1 :         // \J1lTERROUTI CALL NE
```

Subprogramme 'I UPR01 should be loaded from floppy disk and **called** at **address B000H**

Possible errors:

7SYNTAX ERROR	- No filename specified - File name not in citation11gsstr1c/"a included - Nact, 'name' no time Jenen de {RETURN) or Kmman doseparer ":,
?FILE NOT FOOND	- No programme with the specified name could be found on ***the*** disk.
?FILE TYPE KISNATCH	- On *the* floppy disk a file1 of *the* same name **was** rounded, but this is not a machine programme (data type = B) .
?DISK 1/0 ERROR	- e1n failed when reading from the floppy disk. (floppy or centring problem)

BRUN

load and start a machine programme

BRUN "nane'

nane" = maximum 8-digit programme name,
enclosed in quotation marks,

Registered as Direct111111Rando and in programme mode.

A machine programme stored under **the** filename **name"** on the ü1skette loads and runs 1n of memory.

The start of the programme is done exclusively at the programme's initial address (see BSAVE).

Example:

FIFFI BURN

The machine programme 'FIFFI' is loaded and started.

Possible errors:

?SYNTAX ERROR	– No filename specified – Filename not ITI quotes included – After ' name ' no end of line (RETURN) or KOll!llando separator ':'.
FILE NOT FOUND	– Cannot find a Progran m ıtttt in the specified name on the floppy disk.
?FILE TYPE I'IISI'IATCH	– On **the** floppy disk a file of **the** same name **was** found, but this is not a l'laschinenprogramme (File Type= Bi.

?DISK 1/0 ERROR — There was an error reading from the floppy disk. (floppy or centring problem)

RENAME

Rename Files and Programmes

IEN 'naae1', 'naae2'

nael" = file/programme name, old, up to 8 digits, enclosed in quotation marks.

Nah?' = File/programme name, new, max. B8 digits, enclosed in quotation marks.

Authorised as direct command and in **programme todus**,

A programme or file under the name of namel on the disk will be renamed 'name2'.

Example:

REN 'OTTO,' ANTON

The file "OTTO" will be renamed "ANTON'.

Possible errors:

?SYNTAX ERROR
- 'name1' and/or 'name2' are missing.
- 'narnet' or 'name2' does not appear in quotation marks
- Names not separated by coma,

?DISK WRITE PROTECTED	The disk's scream notch is superglued.
FILE NOT FOUND	The file labelled 'namel' does not exist on the disk.
?FILE ALREADY EXISTS	The one with name? labelled file is already present on **the** disk.
?DISK IiO ERROR	An error occurred while reading or writing the table of contents.

DCOPY

Copy a programme
Syntax: **DCOPY "naae"**

'naw,e' = maximum 8-digit programme name,
enclosed in quotation marks.

Only as direct konando allowed1g.

The DCOPY-KOllltando mit specification of a programme name causes this programme to be copied from one disk to another.

After entering the command, the S1e is first prompted to specify the output and destination drive.

SOURCE DISK (1/2)?
DESTINATI<"f DISK U/2>?

Answer each of these two questions by pressing the **'1** or '2' key,

If you only have one drive, **it** will answer any question with '1",

With CTRL/BREAK you can cancel the K0111mando version.

After the drive selection process begins copying. Copying is done by calling the LOAD and SAVE routines, as they are also used in LOAD and llLOAD, or in SAVE and RSAWE,

For this reason, **the** DCÖPY command can't copy a single data file!:IP = D) as **it** is structured differently.

Copy to only e1ne111 drive lSOURCE DISK = DESTINATION DISK), vOr the loading process will prompt

INSERT SOORCE DISKETTE
<PRESS SPACE HEN READY)

and before writing d1e prompt

INSERT DESTINATION DISKETTE
(PRESS SPACE WIEN READYI)

(S0URCE = source, DESTINATION = destination l,

If you have inserted the correct floppy, press the space bar to continue the function.

You can cancel the copy at any time by pressing the BREAK button.
Do S1e h1es during the writing process" so please note the **instructions** at SAVE and BSAVE,

When the copy is complete, the **READY** extension appears.

Example: (with ')' are system outputs)

```
>READY
DCOPV "El'!IL"
>SOURCE DISK (1/21?

>DISK DISK 1/2)7 1
>INSERT SOURCE
DISKETTE ><PRESS SPACE
WEN READY) Space
```

, charging

```
>INSERT DESTINATION
DISK )(PRESS SPACE WHEN
READY} Space

• Write

>READY
```

The **Progra11111** to be **copied** overwrites its original memory area in RAt,

When the copy is complete, regardless of a previous 1gen **DRIVE command**, drive 1 is **always** selected.

Possible errors:

?ILLEGAL DIRECT	Attempted to invoke the DCOPY command from a Progra11111.
?SYNTAX ERROR	'**naael**' not enclosed in quotation marks.
?FILE NOT FOlJND	The programme 'namel' does not exist on the source disk.
	Attempted to copy my data file.
?FILE TYPE MISl1ATCH	The write protection of the target disk is superglued.
?DISK WRITE PROTECTED	There is already a programme on the target disk named name!' is present.
?FILE ALREADY EXISTS	The target disk contents directory is full. The programme can no longer be entered (max, 128 files/programmes),
?DIRECTORY FULL	There is no more space on the target disk.
?DISK FULL	

1DISK li0 ERROR	Scr, write, or read error on one of the two floppy disks.

ERASE

Delete the file or programme on the disk.

Syntax **ERA 1naae**.

name" = maximum 8-digit Progra111111 or file name, in lead time.

Accepted as direct command unc:1 in programme mode.

E1n with name labelled programme or **a** data file1 was deleted on the disk.

To this end, the entry 1m Table of Contents1 is deleted and all sectors enclosed by this file are released:

Example?

ERA 'DAT1'

The file named 'DAT1' will be deleted,

Possible errors:

7SYNTAX ERROR	– no data name specified – File name not in quotation marks
7DISK RITE PROTECTED	The write protection notch of the Dskette 1st superglued.
FJLE NOT FOUND	The **specified** file 1st is not on the floppy disk.
?DISK 1/0 ERROR	write or read errors on the disk,

Saving and Processing Data

File organisation and access

The LASER-DOS he !allows you to store data on the disk from a .BASIC programme t,er and then process it again.

This data is stored by the 1n special data files with the type flag "p.

The 5peichertorm offered is 'sequential'. Sequential means that the data is 1n of the file; 1 bit of pink cartridge, stored in a row. Reading or writing data always starts at the beginning of the file, and further reads and writes *to the* subsequent locations of the file,

In contrast, the direct access type {random access) is used to access any data in a file directly. Unfortunately, the LASER-IÜS does not support this Spe1cher form by default. However, it can be easily replicated with some assembler/machine language knowledge and the help routines described in the last chapter.

Sequential access is data flow-orientated, meaning the number of characters for a write or read operation can vary. You can also talk about variable length data sets, bl0be1 a record the sum of data elements 1st, which are written to or read from the disk with a write or read call.

Sequential files are the simplest form of data storage and recovery. They are ideal for storing unformatted data without giving away much space between the individual data elements. Data is read again in the same order as it was written.

To access a data file, it will be opened previously. A special C>PEN call is available. In this case, you also specify the type of access, whether data should be written or read.

After the data manipulation has stopped, each data file *should* be closed with a CLOSE call.

When editing data files, you should consider some important points:

- If a file that does not exist is opened for writing, it is rewritten and placed at the top of the file.

- If an existing file is opened for writing, the file is placed at the end of the file, i.e. the file is expanded at the following write calls,

- If you want to rewrite an existing file from the beginning, you must delete it first.

- Reading begins after opening 1mm at the beginning of the file. If you are looking for data within a file, you must overread the data before it.

- To update a sequential file, read the source file and write the updated data to a new file.

- Leg Reading the file,$ must be aware of the exact structure of the record to be read, This does not refer to the length of the record and each item; However, you must know the number of elements and the format of each element (string, integer, etc.) and provide a corresponding receive field for each element.
- Within the file, the data is stored exclusively in ASCII format. The elements are separated by commas.
  For example, the number 1,245 takes up 8 bytes of space, including a space for the sign at the beginning and another space at the end.
  The text "RORERT MAIER' claims 1? bytes on the disk,

- A data set should not exceed 208 bytes in length, otherwise difficulties with the internal data structure of the BASIC occur when reading.

  The 2088 bytes count

  - each character
  - the comas as element separator
  - for numbers, the sign and
    an additional space

  a final RETURN (CR) at the end of the sentence

- You can open two files simultaneously, the access types can be the same or mixed.

Attention

If your system should log in with DISK-BASIC V1.0, you should only work with one file at a time. Two open files are still being managed there, and can cause significant data loss.

– The DOS does not tell you when the end of the file is reached when you read it. You have to set this yourself, in which, for example, you write a particular discovery as the last one in the file.

Sequential Output Sample:

A table with data for the conversion of English into 111etric units shall be stored.

English Unit	l'lethal unit
1 inch	2.54001 ca
1 l'lile	1,0935 km
1 acre	404.86 sq m
1 Cubi c Inch	0,01639 1tr
1 U.S. gallon	3. 785 ltr
1 Liquid Quart	0.9463 ltr
1 lbs	0. 45359 kg

The data should be structured on the disk as follows and entered in a data file called 'ENG}MET;

"Unit Engl.> l'letr. unit", conversion factor

e.g. 'IN->Cl'l',2.5411101

The following programme creates such a file.

```
10 OPEN "ENG>l'IET',
1 20 FOR 1% = 1 TO 7
30 READ ES,F
40 PR "ENGMET',ES,F 58
NT
60 CLOSE "ENG>l'IET"
 70 DATA "IN-}Cl'l',2.54001,"l'II->Kl'l',1.b0935,'ACRE->QKM",4046.86E-6
```

```
80 DATA 'CU,IN->LTR',1.638716E-2,'GAL->LTR',3.785
90 DATA 'LIQ.QT-LTR',0. 9463, 'LR-KG' ,0. 4535%
100 END
```

Row 10 creates the file 'ENG>MET' and opens it for writing, In row 40 the first row rn clie file is written.
Line closes the ENG}MET file again.

Sample Segment Input:

The following programme reads the file "ENG>MET" in two parallel matrices and then asks for conversion problems.

```
10 CLEAR 1000
20 DIN ES(),F6)
30 OPEN ENG>MET,0 40
FOR 17. = 0 TO 6
50 INN "ENG>MET' "ES(IZ),FIX)
60 NEXT
70 CLOSE 'ENG>MET'
108 CLS: PRINT ' CONVERSION ENGLISH=>METRIC" 110
PRINT: FOR I:Z = 0 T0 6
120 PRINT TARA); USING "#) Y.              ''; II,E$I7)
130 NEXT
140 PRINT 3320, 'WHICH INVOICE (0-6J'i 150
INPUT VI: IF VI } THEN 190
160 INPUT 'ENGLISH VALUE' iV
170 PRINT "THE METRIC VALUE IS" VF) 180
INPUT 'NEXT NIT <RETURN>"iX
198 G0T0 100
```

In line 30, the file is opened for input. Reading starts on, file start.
In line 50, a record containing the elements E$ {unit) and F (factor) is read and distributed to *the* matrices.
Note that the list of variables is the same as the list of the Schrelo command in the previous programme when it reads.
In line 70 the file is closed again.

Updating a File

If you want to write an existing file by one or more sets of 111e1tern, write d1This file to write and simply enter additional records with PR#, d1e to be appended to the existing data set.

If you want to change data within a file, toigenoes procedure is recommended (not **With** DISK BASIC V1.).

1, Open the file to edit for reading.
2. Open a second new file for writing
3. Read a sentence and edit the data
4. Write **the** sentence to the new file
5. Repeat points 3 u. until end of file · Close **both** files
7. Delete the source file
8. Rename the new file to the source file

With the DISK BASIC V1, only the solution remains to read the file to be edited completely in the memory, to edit and to write completely in the new file. However, the li11indicates the size of the file to the available memory.

OPEN

Open a File

Syntax: **OPEN · naae · , n**

nae' = Maximum 8-digit data name 1n quotes included.

n = Type of access
0 - Read
1 - Writing

Only i11 Programme-Nodus approved.

With **the** OPEN command, a data file {type = D) is opened for writing to read.

The OPEN-K0111111ando internally creates a file control block for each opened file, which contains function codes and pointers.

Also 111is positioned according to the access code to the file:

- When reading 1mm to the top of the file

- When writing to a new file to the top of the file

- When writing to an existing file to the end of the file.

Since only two file control blocks are provided in the system, only two files can be opened at the same time. The access mode is dabe1 otne relevant, both can be opened for writing, both for reading the n zun in and the second for writing (see limitation DISK BASIC VI.I on the previous pages).

Example:

OPEN TEST,0

The data file "TEST' 111is opened for reading.

A data file can only be opened once at a time. Attempting to open the same file still11als will result in an error message.

Since *the* file control blocks (FCB) are outside of the BASIC programmes, a file remains open if the calling programme is aborted before the CLOSE call due to error or pressing the BREAK key 11JLtrde and maybe not already in memory. Such a file cannot be opened easily,

If a BASIC programme is cancelled 111,
without neatly closing its files, so you should catch up with it via direct account CLOSE "Filenade".

Possible errors:

?ILLEGAL DIRECT	Attempted to run the OPEN command in direct l'lodus.

?SYNTAX ERROR	– one or both parameters are missing – no comma as separator – **•name•** not quoted – Access type not or 1
'FILE ALREADY OPEN	File is already open, close with Direct011t111anclo 'CLOSE' if necessary.
'FILE TYPE KIS1'IATCH	The file addressed in the OPEN-K01111ando is not a data file
FILE NOT FOUND	A file to open for reading does not exist on the disk.
?DISK BUFFER FULL	There are already two files open and there is no longer a file control block available.
DISK I/0 ERROR	My error occurred while reading from the floppy disk.

PR#

Write records to a file

Syntax: **PRt •naae• List of elements**

' nane = maximum 8-digit **filenames**, enclosed in quotation marks.

Item List= List of variables and values to write to file. The individual elements are separated by commas,

Mur in programme mode,

Collects a record from the values of the item list and causes it to write to the data file.

This must be opened with an OPEN command for writing.

Be1spiel'

```
Z00 A1 = -40.456: B$ = 'STRING VALUE'
210 OPEN 'TEST'!
220 PRAN 'TEST',A1,B, 'THAT'S
IT' 230 CLOSE 'TEST'
240 END
```

After opening the file 'TEST 1n line 210 is compiled in line 220 emn record and 1n this file is written.

The record contains the current values of A1 and Bf and additionally the string "DAS WAR'S'. The values can be read back later with an IN# command.

1st, make sure that the number and type of elements in the IN# command 1n's list of eggs is equal to the PRN command,

The values represented by the list of elements should not exceed 200 characters in total. This includes not only the values themselves but also all the separator characters {Komas) between the values, at nu111er1scr,en values additionally the sign digit and a closing space and finally the record end identifier (CR).

The previous example of record 1m would have a length of 31 characters

```
        -40.456 ¡STRING, THAT'S IT
```

Unfortunately, when creating the list of elements, you often do not know exactly how large8 the individual variables will be at the time of storage, then only cautious estimation helps. Always stay on the safe side and divide your list of elements into several PR#commands.

Unfortunately, the PR# command does not notice when a record becomes too long. D1eser is simply written 1n full length on the disk, problems then make the reading in with dell IN# Koll!lllando, 1J10be1 in the simplest case 'only' data will be lost.

Writing a record does not necessarily cause in physical writing to the file. Only full sectors are written to the disk. The PR# command data is collected in from an internal buffer that is the size of a sector, Whenever the Buffer is full, it is transferred to a free sector on the disk and then a new free sector is identified, this writing of a sector can be done in the middle of a PR# command; it may also require several PRll-K011t11tandos to fill a sector.

A lllit PRI to be written record 1111rd written without regard to sector boundaries. l'lan also refers to the sector as ph!,fsical unit1 t, while a record represents a logical unit.

l'possible errors:

ILLEGAL DIRECT	Attempted to execute a PR# command directly.
?SYNTAX ERROR	- No filename specified - File name makes in quote - no item in the list - no comma as separator
?FILE NOT	File not opened previously.
OPEN ?ILLEGA	The file was opened for reading.
L WRITE	The write protection notch on the floppy disk is superglued.
?DISK WRITE PROTECTED	No free sector was found on the floppy disk.
DISK FUI.L	Leg reading or writing occurred on the floppy disk, 1 it.

?DISK I/0 ERROR

If one of these errors occurs, the a1t programme of the corresponding error message is terminated. Please note that this file was not closed afterwards, you should do this manually.

IN#

Import records from a file

Syntax? **1Nt maue,Eleentenl iste**

'name' = maximum 8-digit file name, enclosed in quotation marks.

List = Load of variables to read ~from the Date1. Separate the variables by comma.

Only i111 Programme-l'lodus approved.

IN# reads a record from the specified file and assigns the elements of that record to the specified variable;

The file must previously be opened by an OPEN-Ka.ando ZWII Read 1110rden.

Example:

```
28 OVEN 'TEST,0
210 IN#
'TEST',X,A$,B$ 220 CLOSE
'TEST'
```

This example refers to the record created in the example of the PR) command in the file 'TEST. The data stored there are assigned in turn to the variables of the INt-Ko11111ando.

After running the row 210, the variables contain the following values:

X= —40.456
A$ = 'STRING VALUE
B$ = 'THAT'S IT'

The list of elements of the IN# command must match the list of elements of the PR# command in terms of the number and type of variables. Also 1st in order be1 to keep the T!:jpen, the naming is red,ne relevant.

If IN# is used to continuously read sentences from a file, it is difficult to know the end of the file at the right time. There is no special 'END OF FILE' identifier in the LASER-DOS.

There are several possible solutions:

- the number of sentences is known, they are counted 1m reading programme m1t one counter,
- a second small file contains the record counter for d1e main data.
- a short label consisting of only one alphanumeric corpse {2.R. PRN 'name', · A').

  In the read-out programme Il!This pass phrase is read before every read of a record, e.g. IN# name,A$)
  If the receiving string variable is then empty, the end of the file erre1 is eight.

MayJ 1che Error:

?ILLEGAL DIRECT	The INIl command was entered as a direct command,
?SYNTAX ERROR	- No Filena111e specified - File name not in quotation marks - no item in *the* list - no coma as separator
	The specified file is not open.
7FI LE NOT OPEN	The 1111.1rde file *opens* for writing. There *was* an error reading from the floppy disk.
/ ILLEGAL	
READ ?DISK I/(J	
ERR<JR	

	One *of the* specified variables does not match the data read from *the* disk by type ner. The programme continues, the variable remains empty,
7EXTRA IGNORED	In the variable list of the IN# command, wem1ger variable are specified as values 1111 record are present, which are above-. Numerous values are ignored, the Pro-9ramm continues,
	The variable list contains more variables than values ıılrecord are present. The programme he111Now starts typing the missing values via the keyboard.

If any of these errors occur on iau~r REDO, EXTRA IGNORED and ??), the programme will be terminated after the corresponding message is issued. Please note that as} d1se file has **not** been closed, you should do this manually.

CLOSE

Close a Data File

Syntax: CL.OSE **"naae •**

nane' = maximum 8-digit file name, enclosed in quotation marks.

Approved as direct KONando and in programme mode

tit de CLOSE command closes a previously edited data file.

If a file or file is opened for reading (i.e. the last file access was not to this file) or in direct mode, only the file control block (FCB = File Contro! block) again. Diskette access does not occur.

However, if the CLOSE command is given in programme mode and the file to be closed is open for writing and is currently active, then the last sector in the putter will be written back to the floppy disk so that no data will be lost.

It is good programmer's use to close every open file after use. However, it is essential for output files unless you accept data loss.

Example

```
CL OSE 'MAILBOX'

The data file 'MAILBOX' is closed.
```

The closing and reopening of a file is required even if you want to change the access type (e.g. from write to read).

If the file to close is not open at all, i.e. there is no open file control block for this file, the CLOSEtonando will be passed without any error message. This is especially useful for closing all files used in a PrograH prophylactically at the end without checking 111which is currently open.

Possible errors:

?SYNTAX	– No filename specified – File name not in citation sign.
ERROR	The write protection notch on the floppy disk is superglued.
2DISK RITE PROTECTED	An error occurred while writing to the floppy disk.
DISK I/0 ERROR	

Summary of possible DOS error messages and their probable causes.

?DIRECTORY FULL	Attempted to run a programme or save a file from the floppy disk containing berel ts 120 entries.
?DISK BUFFER FULL	Attempted to open a file with OPEN, although two files are already open.
7DISK FULL	There is no free sector left on the disk.
7DISK I/O ERROR	There was an error writing or reading leg. e.g. address mark not found, check points wrong, etc.
?DISK WRITE PROTECTED	Attempted to write to a disk with a write-protect notch glued over it.
?FILE ALREADY EXISTS	A programme to be stored on the floppy disk is already included there,
?FILE ALREADY OPEN	It just makes an OPEN call to a File dropped that is already open.
FILE NOT FOUND	There is no file or programme to load for reading on the disk.
?FILE NOT OPEN	Attempted to **use IM** or PR# to edit a file that **was** not previously opened.

?FILE TYPE MISMATCH	An attempt was made to edit a file of the wrong type. LOAD/RUN - File type unequal • r • BLOAD/RRUN - Data type unequal "B OPEN - File type unequal 'D' DCOY - File type = 'D
?ILLEGAL DIRECT	Attempts have been made to apply a DCOPV kernel mando in root#odus or one *of the* OPEN, IN# or PR# commands in direct mode.
?ILLEGAL READ	The prayer! IN# dropped *the* file for *a* file *that* is open for writing.
?ILLEGAL WRITE	*The* .PRII 1111.1rde command is set for *a* file open for reading.
?INSUFFICIENT MEMORY FOR DOS	Attempted to initialise the DOS system on *e1nem* LASER 110 or VZ200 without memory expansion.

1. As mentioned in the description of the last sections, the problem exists, since all files will not necessarily be completed correctly when a programme is cancelled by error or BREAK key.

A restart of such a programme after error correction or similar will usually result in the message 'FILE ALREADY OPEN.

You can now manually close these files in direct l'lodus if their names are known.

However, this is not enough for newly created files in a 111 Progra11111. Such files must then also be deleted, otherwise
if you open again, the file will be rewritten and you will have your data in the file several times.

In all these cases, the following procedure is recommended:

For all the programmes in development, you define
at the end a block with CLOSE calls and, if necessary, delete calls for all files addressed in the programme. For a .a. ProgrammeInterruption then simply call this routine with **RN line** number.

Example:

You will edit in a programme the three files DAT1, DAT2 to read and DAT 3 will be recreated,

```
        own programme

4800 END

20000 CLOSE 'DAT1'
20111 CLOSE 'DAT2"
200820 CLOSE 'DAT3'
20038 ERA 'DAT3'
20040 END
```

If a programme interruption occurs, clean your files with RUN **Z88** and can restart your programme without any problems after correction.

2. Often it is necessary to have my file present on *the* floppy disk when *the* programme starts (see programme example 'Anschri ftenverz.' **Yes**, although it does not contain any data.

Apply a procedure similar to the one above by defining the following lines at the end of the actual programme:

```
        own programme

6000 END

1000 OPEN 'MAILBOX,1
108010 CL0SE 'MAILBOX'
10020 END
```

With **RMN** 188@ S1e create an empty data "MAILBOX on the chain.

3. A bottleneck **of the** LASER-DOS is that the filenames in the 111**d**'s **comman** must be specified and cannot be replaced by variable,

How can 111an still handle different files in a programme.

Kemntm1sse of the BASICProger structure is required to understand the solution provided below.

The main points are:

- The initial address of a BSI programme can be found in the 78A4H and 7845H 4320884 u locations. 30885 decimal).

- A BASIC line has the following structure:

```
2         - Move to the next row
bytes 2   - line number
bytes n   - line text
bytes 1   - End-of-Line ID (X'00')
bytes
```

In line text, RASIC keywords, other than the DOSKOMandos, are represented in the text as single-digit 'TOKENS'.
The space inserted between the line number and the line text in a Pl"'ograma listing is not part of the line.

Taking these conditions into account, the example can be easily understood and replicated.

The key point of this example is that all file calls are at the beginning of the programme, so they are more countable and will not be moved in the case of later programme changes.

Example

A programme evaluates one of the possible DAT1, DAT2 or DAT3 files after the user has selected it.

```
10 GOTO 100
28 OPEN 'DATA' ,0:RETURN
3 INN 'DAT1', A$, B$, C:
RETURN 40 CL0SE
'DATA' :RETURN

1080 CLEAR 1000
110 A=PEEK(30885)f256+PEEKi30884)
120 CLS
130 INPUT 'FILE VERSION (1-
3)';X$ 14 IF Y$'1' 0R Y$33° THEN
128 150 POKE A+23,ASC(X$)
160 POKE A+42,ASC!X$)
170 POKE A+9.AS€X$)
180 GOSUB 20
190 GOSUM 30

        edit the data, if necessary read several
        sentences ad t GOSUB 30.

400 GO5UB 40
410 END
```

Row 10 jumps to the beginning of the programme.

Lines 2, 3@, and 40 define the file views as single subroutines.

In line 11@, the programme start address is determined.

Lines 13 and **14** request the desired file version.

This 111ird, IIIE!nn correct, in rows 150, 160 and 170, is transferred to the filenames of rows 20, 30 and 40.

For example, in lines 180, 190 and 400, file manipulation is indicated by subroutines.

• Three-Run Application Sample

The 'Address Management' programme shown on the following pages shows a typical example of floppy disk editing using the LASER-DOS.

It allows the creation, storage and processing of up to 1 addresses.

The Aoressen are stored on the floppy disk in a file called 'MAILBOX'.

The editing procedure is chosen only de with regard to the weaknesses of the DOS BASIC I.II, so that the file contents will be completely read in the memory when the programme starts and at the end of the editing was completely written back to the disk when changes were made.

Operation of the Pr Lograns

The programme will be loaded and started with RUN "ANSHR". Immediately after
Start will transfer the contents of the file 'MAILBOX' to programme internal files. This file must be located on the disk, otherwise the programme will be terminated after an error message is reported.

After loading 1111rd the menu is output

(1) NEW ENTRY
(2) UPDATE ENTRY (3) READ
ENTRY
4) READ ENTRY
(5) SORTED LISTS 16)
QUIT PROGRAMME

You select one of these functions by entering the corresponding digit.

After completion or functions 1 to 3, these are self-explanatory, the programme returns to menu output,

When function 6 is completed, the programme stops:

If the data content was changed during the programme run, the addresses
Required before performing the functions 5 and 6, 111enn, sorted alphabetically by surname and first name.

In function *b*, the data is written back to the disk when changes occur.

The programme expects the Date1 MAILBOX to be present at startup, and 31e will apply the programme for the first time; so 1st there is such file on the disk, hit the following procedure you can create an empty data "M]LROX on the disk:

```
LOAD
'ANSCHR" RUN
3000
RUN
```

This creates a file 'MAILBOX' and then starts the actual main programme.

Zua Prograabau

~~The programme has a~~ modular structure; i.e., every Funktlon is realised in a closed routine.

After **dell** Start, the file 'MAILBOX' is read in first (lines 220 - 280).

Note that this file has a solution selected for final recognition.
be1 of the before any 'real' data record a key record with a short alphanum. text is saved. The reading and evaluation of the passphrase takes place in cells 240 and 258, the reading of the 'real' sentence in line 280.

On the individual programme routines for address processing (functions 1-5) should These have nothing to do with de111 DOS directly.

You can analyse *the* routines yourself if necessary.

Only one Hin111eis '. To 1 Sort the addresses (rows 2200 - 2390) was added to the "SIELL' SORT - procedure!This is somewhat more complicated by the structure, but considerably better than the simple and usual "BUBBLE" SORT **in** terms **of** runtime.

The data is written back to the disk in the rows 1880 to **1920.**

For security reasons, the data will be written to a temporary file 'TEMP. The old file 'MAILBOX' 1111rd is deleted and then the temporary file is renamed 'MAILBOX'. This has the advantage that if errors occur during write (DISK F\JLL or similar), at least the old file still exists.

If you are interested in the programme, just tap it and save it on the floppy disk with **SAVE 'ASCHR.**

```
!00
'*********************************************
110 '*
 120 '*        ADRESS MANAGEMENT
130%
 140
'*********************************************
 150 °
160 CLEAR 2000
 170 HD$='     ADDRESS LIST'
180 DIM NN$(99),VN$(99l,TI$(99),ST$i99),NR$(99),PL$(99i,OT$(99i
200'AJ,jFONT LIST READ
210 GOSUB 2450
Z2O OPEN 'MAILBOX' ,
230 FOR N = 0 TO 100
240 IN# "MAILBöK,As
250 IF AS    'THEN 280
2@ IN# 'MAILBOX' MN$(N}, WN$N),TI$(N),S7$(N),NR$N),PL$N),0T$(N) 270
NEXT N
280 CLOSE 'MAILBOX'
300 'SPEND MENUE 310
GOSUB 2400
320 PRINT
330 PRINT TAB(4li"(1) NEW ENTRY'
340 PRINT TAB(4);'(2) UPDATE ENTRY' 350
PRINT TAB(4);'(3) DELETE ENTRY'
360 PRINT TAB(4li"(4l READ ENTRY'
370 PRINT TAB(4ii"(5) SORTS LISTS' 380
PRINT TAB(4l;"(6l EXIT PROGRAMME' 390
GOSUB 2500
400 A$=INKEY$: IF AS< '1' OR AS> '6' THEN 400
410 IF AS= 'I' THEN 500
420 IF A$ = '2' THEN 700
IF 430 = '3 THEN 1200
IF 440 = '4' THEN 1300
450 IF AS= '5' THEN 150111
460 IF SO= 1 GOSUB 2200
IF MO = 1 GOTO 1B00
480 CLS: END
500 'NEW ENTRY 510
GOSUB 2400
520 IF N = 99 PRINT 'FILE MAILBOX ALREADY FULL': GOTO 2450
530 INPUT "LAST NAME    ';NN$: IF NN$ = …THEN 510
540 INPUT "FIRST NAME   'iVNS
550 INPUT TITLE         '; TI$
560 INPUT ROAD          ';ST$
```

```
570 INPUT HOUSE NUMBER ";NR
580 INPUT
'POSTLEITZAHL';PL$ 590 INPUT
'ORT ' ;on 600 GOSUB 2001!
610 IF GF = 1 THEN 2550
620 MN$(N)=NN$: WN$(N)=WN$: TI${N)=TI$: ST$(N)=ST$: NR(N)=NR
630 PL$(N)=PL$: OT$(Nl=OT$
4 N = N + 1
65 MO= 1: S0= I
660 PRINT: PRINT 'ENTRY PERFORMED' 670
GOTO 2600
700 'UPDATE ENTRY 710 GOSUB
2101!: GOSU!.I 2000 720 IF
GF = 0 THEN 2700
730 GOSU! I 2401!
740 PRINT '1, SURNAME: 'NN$(ll
751! PRINT'2. FIRST NAME: ';VN$(!)
760 PRINT '3. TITLE: '; Tl$(!)
770 PRINT '4. ROAD: ';ST$(Il 780
PRINT '5, HOUSE NO.: ";NR(!)
790 PRINT '6. Postcode: ';PLSI'

800 PRINT '7. '10T$1)
LOCATION: 811!
GOSU, I 2501!
A$820=IMKEY$: IF A$ €" 0R A4$ } "7 THEN 820
830 IF A$ = '0' THEN 301!
840 IF A$ > '1' THEN 890
850 INPUT 'NACHNAME 'NN$
860 IF NN$ = ' ' THEN 730
870 IF NN$QNN$I) THEN NN$)=MN$: S0 = I: MO = 1 880
GOTO 730
890 IF A$ > '2' THEN 930
900 INPUT 'FIRST
NAME' ;WN$
910 IF VN$0VN$(I) THEN VN$( I l=VN$: SO = 1:M0 = i
920 GOTO 730
930 IF At> '3' THEN 970
940 TITLE INPUT ';TI$
958 IF TI$QTI$(I) THEN TI$(I)=TI$: NO = 1
961! GOTO 730
970 IF A$ > '4' THEN 1010
980 INPUT 'STREET' iST$
998 IF ST$OST$I) THEN ST${I)=ST$: HO = 1
1000 GOTO 730
1010 IF A$ > '5' THEN 1050
1020 INPUT 'HAUSNUMl'IER';NR
```

```
 103 IF NRQNR(I) THEN NR(I)=NO: MO = 1
 1040 GOTO 730
1050 IF AS > '' THEN 1090
10690 INPUT POSTAL CODE ';PL$
 1070 IF PL$(}P1$(Il THEN PL$(I)=PL$: MO=
 1,1080 GOTO 730
1090 INPUT "ORT ';OT$
 1108 IF OT$0T$(I) THEN 0T$(I)=OT$: MO = 1
 1110 GOTO 730
1200 'DELETE ENTRY
 1210 GOSUB 2100: 60SUB 2000
 1220 IF 6F = 0 THEN 2700
 1230 PL$1Il = ' XXXX '
1240 PRINT: PRINT "ENTRY DELETED" 1250
MO = 1: 60T0 200
1301il 'READ ENTRY
 1305 IF SO = 1 GOSUB 2201r'J
 1310 GOSUB 218111: GOSUB
 2000 1320 IF GF = 0 THEN
 27011 1330 GOSUB 2400
1340 PRINT T1$(!)
 135 PRINT WNS(IY 'ANS(I) 130
 IF ST$(!)=' THEN 1390 1370
 PRINT ST$(I1;
1380 IF NRI) = PRINT ' ELSE PRINT NRII)
1390 IF Pl.$(1) = ' THEN 1395 ELSE PRINT P1$(I1" ';
1395 PRINT OT$(11
1408 I = I + 1: G0SUB 118
1410 IF IN THEN 1330 ELSE 300 1500
'ISSUE LIST
1510 IF SO = 1 GOSUB 2210
1520 I = 0
1530 GOSUB 2400
1540 IF N = 0 PRINT "NO ENTRIES AVAILABLE" : 60T0 2600 1558
FOR J = 1 TO 12
1560 IF I = N THEN lb08
1570 IF P1$(I) = ' XXXX ' THEN
1590 1580 PRINT NN$( I l ',' VN$(1)
1590 I = I+1 : NEXT J
1608 GOSUB 1610: IF I <N THEN 1530 ELSE 300 161
PRINT i-480,'<RETURN> = NEXT, <E> = END'; $1620 =
INKEY$
$1639 = INKEY$: IF A$ = "" THEN 13 164
IF A$ = E' THEN I = N: RETURN
```

```
1650 IF A$ <> CHR$(13) THEN 1630
1660 RETURN
1800 'WRITE DATA ON DISK 1810
GOSUB 2450
1820 OPEN 'TEMP',1
1830 IF N = 0 THEN 1890
1842 FO I = 0 TO N-1
1850 IF PL$(li = ' XXXX ' THEN
1880 1860 PR# 'TEMP','A"
1870 PRN 'TENP',NNSI ,WN$(H,TI$(I,ST$(I),NR1,PL$1),0T$(I) 1880
NEXT I
1890 CLOSE 'TEMP'
1900 ERA
'MAIL.BClX'
1910 REN 'TEMP','NAILBOX'
1920 CLS: END
2000 'FIND IN LIST 2010 FOR I =
0 DOOR N-1
2020 IF PL$(J) = ' XXXX ' THEN
2060 2030 IF NN$ } M$(I) THEN
2060 204 IF WN$ = ' THEN 2070
2050 IF VN$ = VN$(Ii THEN 2070
2060 NEXT I: GF = @:RETURN
2070 GF = 1: RETURN
2100 'READ SEARCH CRITERIA
2110 GOSUB 2400
2120 INPUT 'LAST
NAME' ;NN$ 2130 IF MN$ = '
THEN 2110 2140 INPUT 'FIRST
NAME ' ;VN$ 2150 RETURN
2200 'SORTING THE ENTRIES 22108
G05\U8 2450: S0 = 0
2220 IF N < 2 RETURN
2230 N = N
2240 N= INT (M/2): IF M = 0 RETURN
Z25 J = 1:K=N-M
2260 I = J
2270 L = I + M
2280 x= I-1: Y=L-1
2290 IF IffIm < l'ffl(Yl THEN 2390
2300 IF NNSm > NN$(Y) THEN 2320
2310 IF VN$(Xl <= VN$(Y) THEN 239 0
$2328=MN$(X): WN$=W$X: TIS=TI$(X): ST$=ST$(): NR=NRX) 2330
PL$=PL$X): 0T$=0T$(X): NN$)=Ms(Y): WN${X)=Ws(Y)
2340 Tl$!Xl=TJ$(Y): ST$(Xl=STS!V): NR(X)=NR(Yl)
```

```
PL$2350(Xl=PL$(Y)): OT$(X/=OT$(Yl: NN$(Y)=NN$
 236 WN(Y)=WN$: TI$(Y)=TI$: ST$(Y)=ST$: NR(Y)=NR
 2370 PL$(Yl=PL$: OT$(Yl=OT$
2380 I = l - M: IF I} O THEN 2270
2390 J = J + 1: IF J )- K THEN 2240 ELSE 2260
2400 'HEAD LINE
 2410 CLS: PRINT HD$: PRINT: RETURN
 2420 'PLEASE WAIT
240 CLS: PRINT 3228, 'H#+#s THIRD WAIT #+s' 2470
RETURN
2500 'DIGIT
2510 PRINT: PRINT 'PLEASE ENTER NUMBER i0=END)': $2520 PRINT
= IMKEY$
2530 RETURN
2550 'PRESENT ADDRESS NOTICE'
250 PRINT: PRINT "ADDRESS ALREADY AVAILABLE"
2600 'WAITING FOR RETURN
2610 PRINT: INPUT 'CONTINUE WITH
<RETURN>';X 220 60T0 300
2700 'ADDRESS NOT THERE NOTICE'
2710 PRINT: PRINT 'ADDRESS NOT AVAILABLE' 2720
GOTO 2600
38080 OPEN 'MAILBOX',I
3010 CL0SE 'l1AILBOX ·
31120 END
```

Diskette setup and organisation

Diskette setup after initialisation

Before working with a disk, it must have the basic structure *of the* tracks and sectors.

This basic structure is written to the floppy disk using the INIT command.

It consists of 4 tracks a **1** sectors with 128 bytes each. As already noted in the section "Usage accounting structure", k011tmt to each sector still a certain "overhead", which consists of not10endigen synchronisat1ons and addressing fields. This results in a total length of 154 bytes per sector.

Such a sector has the following basic structure:

Byte **0** Byte 7	Adress$-3gnchromsatin	7x#'80°
- 10 Byte 11 -	Adrell!	X'FE E7 18 C3'
13	three-way Byte 11 = Track number (0 - 39) Byte 12 = Sector number (- 15) Byte 13 = Checksum "Adrejteld' (Track#+ Sector#)	
basked 14 - 19		
bytes 2 - 23		
bytes 24 - 151	Data Synchronisation *b* x X'80'	
bytes 152 - 153	Data Label	X'C3 18 E7 FE'
	Checksum11111e Data Field	= 128 bytes

In *the* initialisation, each sector is written in its entirety, with data field and Prütsu111111e !Byte 24 - 153) = X'00' set.

The sectors are numbered consecutively around the floppy disk, but arranged in triples (see Figure 1.6). This ensures that continuous sectors of an external track can be reached during a rotation of the disk if they do not exceed certain processing time in between.

This amount is 94 ms from the end of a new sector until the beginning of the next sector appears in *the* numerical order under the letter/reading head. 94 ms is *a* huge amount *of* time you can do extensive data manipulation.

39 traces of the disk are available for storing programmes and data.

The first track of a floppy disk (track) is used *to* manage floppy disks. It contains the contents of the disk and a sector occupancy summary.

The Table of Contents

In the first 15 sectors of the track 0 (Sector **0** - 14) is the contents *of the* disk.

Each entry has a space of 16 bytes.

This results in a capacity of 8 entries per sector and 8x 15 = 120 entries in the **entire** table of contents (see error11message '?DIRECTORY FULL'}).

An entry in the table of contents has the following structure:

B!ite II	Occupancy Status/ File Type **0** – End *of* used entries **in** Table of Contents 1 - shared entry (e.g., after an 'ERA') D - Entry refers to a data file T - Entry **refers** to a text file (BASIC-Progr.l B - entry refers to a binary file (Naschinen- **Progras)**
B!ite 1	Separator li1111 ':')

Byte 2- 9	File Name
Byte 10-11	First sector address of this file Byte l@-Track number Byte 11 - Sector111111er
Byte 12-13	only for file type= T of the initial B channel address 1m memory
Byte 14-15	only for file type= T or B Pr ogr anr end address in Spelher

l'lit den', K0111111ando 'DIR' are simply output, without any preparation, the first 1 bytes of every occupied entry on the screen.

If a Date1 is deleted, the status byte Byte 0) will be set to '1. All other entries remain.

Sector management

In the last sector of the track is the occupancy chart for the sectors of the disk.

For each sector from track 1 there is a bit reserved that indicates whether the corresponding sector is free (bit =) or occupied (bit= 1) 1st.

Be1 39 tracks and 16 sectors per track gives the 62 bits required or 78 1/ytes containing relevant information in this sector.

Be1m Writing a file, this occupancy chart is used to determine the sectors required for storing. Sectors are always used from front to back, filling in gaps that may have been caused by deletion.

Game type 1:

Track 0/ Sector 15

Byte ==> trace 1, sectors 0 - 7
Nged ==> trace 1, sectors 8 _ 15
1 Byte ==> trace , sectors 0 - 7
2

==> · Track 39, Sectors 8 - 15

Good
77

Saving **Profiles** and Files

All programmes stored on the floppy disk will have a corresponding entry in the content directory, with the type of the file or programme being noted in the first byte.

Z111i text files iBASIC programmesJ and binary files (machine-programme!llen) this type identification is the only difference. The recording structures are identical.

The different T!;Jp marking be111has a different treatment after loading bz111. Launching such a programme (see Korr,mando descriptions LOAD/RUN bz111. BLOAD/BRUN).

In the table of contents is in the B!;ltes 10 u. 11 a pointer to the first sector occupied by this programme.

The bytes 12 - 15 of the table of contents contain the information about the storage area of this programme to be transferred when loading. The bytes contain 12 u. 13 the initial address and bytes 14 u. 15 the final address+! of the transmission area.

The data sectors contain in the bytes 0 - 125 of the data field a 1:1 copy of the storage area, i.e. in binary data representation.

The sectors covered by a programme need not be physically sequenced, but can be scattered on the disk. In order to read a programmenevertheless **in** relation to the additions, the individual sectors are delayed among themselves,

In bytes 12 and 1?l of the data field there is a pointer to the next sector of this programme (track and sector number) or '0' in the last occupied sector.

In the case of data files with the type designation ', the contents directory does not contain the information about a storage area to be occupied, the bytes 12 - 15 are irrelevant.

In bytes 10 and 11 also the pointer to the first occupied sector is included.

The individual sectors of the file are indexed as in the programmes.

Each sector of a data file contains in the first 126 bytes of the data field the actual data and in the last two bytes of the data field a pointer to the next occupied sector bz111. **'0'** at the end of the file.

The structure of the data in the first 12 bytes differs from the other two file types.

Data representation is only in ASCII format. The data is stored on the basis of records, with each PR# command writing a complete record to the file 111is.

Records contain a defined end identifier. This is the ASCII character for 'carriage return' X'0D.

A dataset is not sector-specific, it may contain multiple datasets in **a** sector; a dataset may also cover several sectors. Except for the first record of a file, records do not start at a sector boundary.

Within the datasets, the various data fields are separated by commas, **they are** read from each other **by** the Trace 111i variable defined in the IN! command.

In order to process the diskettes, DOS creates various data structures in the last 31 bytes of the available RAN memory, including: edit vectors, file management blocks, and input/output buffer (Figure 1.7).

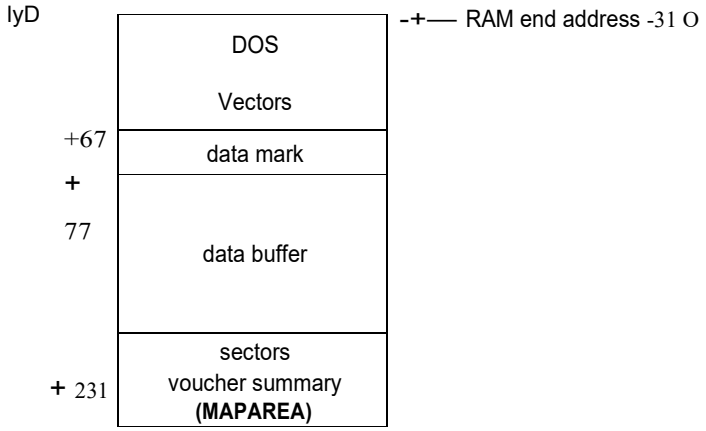


Figure 7,1 The memory areas of the DOS

DS - Vectors

The first 7 bytes of this DOS workspace include the DOS vectors.

The beginning of the DOS vectors is addressed to v011 DOS at the 5!:!st1111 initialisation 111it the ZB88 register 'IY. The DOS expects that this register will not be changed by users per grass, otherwise it will inevitably lead to system crash and possibly also to the destruction of data content on the disk (bitter experience of the author}.

The DOS vector range is structured as follows:

OSVTR = IV

Name	Byte	offset	Importance
FILNO		Y+0	File number. Stent when editing a data file here is the number of the data used1 administrative blocks (iFCB). **0** = FCBl ı 1 = FCB2
th	8	IY+l	File name. Name of the file to edit. Is from If the programme before each file/ Enter the programme access.
TYPE	2	Y+9	File type. Byte 1 = Debit Type. Byte 2 = Actual Type, The user programme is in the first byte specify the type of file to be repaired;
OK		IY+ll	the drive out. X'10' = Drive 1 **'88'** = Drive 2 at initiation, Y'1@' set.
RQST		Y+12	Access type. **0** = Read 1 = Write Is programmed by user to set **For** BASIC, this is done by the OPEN Komando.
SOURCE		Y+13	Output run111erk (source) at DCOPY-K0t111ando 11 or 21
UBFR	2	IY+l't	the address of a user buffer area; to the or from **the** data who should

			When Loading and Saving Programmes 1st is the programme area. When reading data files is it BASIC Output Buffer
DESTIN		I+1	finish at the DCOPY-Kom",ando (1 or 2)
SCTR		Y+17	Number of the sector to address.
TRUE		Y+18	the number of the track to be addressed;
RETRY		Y+19	Replay counter for read errors (Pr for When initialised, **10** set.
DTRCK		1Y+20	the current track number above which the Scroll b/ read head located.
NSCT		Y+21	Favourite field for the next to address- The rotating sector.
NTRK		1Y+22	Favourite field for the next to address- The track.
FC.Bl	13	Y+23	File Management Bl Rock 1. (Structure see *own* description)
FCB2	13	1Y+3	File management block 2. (s1ehe structure own description‡
DBFR	2	IY+49	*Put* the DOS data buffer for writing and reading a sector It is located right in the to the 00S vectors i111 workspace.
LTHCPY		Y+51	Copy of the disk *command* byte ten control.
MAPADR	2	1Y+52	Point to the DOS putter, in the wash the sectors allocation summary **is stored** This is located behind the data

			Buffer in workspace et..
TRKCNT		Y+54	track counter for the DCOPY command.
TRKPTR		1Y+55	track pointer for the DCOPY command.
PHASE		Y+56	pulse step grid for the track adiustment
			Young.
DCPYF		Y+57	DCOPY flag
RESVE	10	Y+5	reserved for extensions.

File Control Blocks = FC

Within the [USS vectors are two 13-byte file management blocks, FCBI and FCR2.

These are required when editing data files to provide status and control updates to the file in access.

The public command determines a free data management view and provides the necessary parameters for the file to open.

The commands IN# and PR# or1enteren sirh at the corresponding file management block, e.g. which sector of the file to read and at which byte of this sector the processing should continue.

From the LOSE command, the de Date1 administrative blocks are released.

Since there are only two of these blocks, only two files can be opened at the same time.

A file management block has the following structure:

FCRI or FCR2

Name	bytes	Importance
FLAG		Displays the status of the FCB. 0 - FCB not used. 1- FCB occupied, file z.lt. makes active 2 - FCB occupies, file active. Active means that currently a current sector of this file is in the data buffer for editing
ACCESS		Access type for this data. - Read 1 - letter
	a	Filena11e
FNAI'I		Spurnunmer
TRK#		Sector1111Aer of the currently in progress sector of the file.
SCTR#		
PTR		Pointer to the next byte to be edited in the above. sector,

Input/ Output Buffer

The DOS workspace contains two buffer zones, one for intermediate storage of sectors to read or write, and a second for the sector occupancy summary;

Data Buffer <DBFR)

This buffer has a 154 byte size and serves as Z111s memory for direct data exchange with the disk.

From data buffers, the sectors are transferred to the disk when writing, and the sectors are transferred from disk 1n *to* data buffer when reading.

Before the data buffer, the 10 bytes *of the* data mark are set at initialisation, so **that** a complete information block (data mark + data field) is available **when** writing a sector.

During normal read/write operations, only the first 128 bytes of the data buffer are used to record a sector's data field.

The full length of 154 bytes is only required during floppy initialisation to accommodate **a** full sector, including all synchronisation fields, adder fields, and tag codes.

Sector Occupancy Overview !NAP)

At the end of the DOS workspace is an 80-byte buffer area, in which the Sector Occupancy Summary from sector 15 of the track is cached from the disk.

When storing a programme or writing a data file, sector selection and allocation are only made to this buffer area after the current sector **has been** initially scanned. Only when the storage process for the **programme** is complete, the occupancy summary is written back to the floppy disk.

The z1111 switch to DOS and floppy control is connected via 4 On/Off 9abe ports, which are in hexadecimal notation ports 1MH, 11H, 12H and 13H.

PORT 18 = Command Tab {0/P LATCH)

via this port the control information is transferred to the command register of the diskette control.

7 bit	=	Drive 2 - Select 1 = yes}
bit 6	=	Access Type (= Write, 1 = Read
bit 5	=	Output pulse when writing to disk
bit 4	=	Drive 1 - Selection (1 = yes)
Bits 3-0	=	Track adjustment step phases

A copy of the port content is kept in *the* DOS vectors in the LTHCPY field.

Initialise this field with 0110 01,

The current step phases are kept and **edited** in *the* 'PHASE' field *of the* DOS vectors.

The selected drive is in the DK field.

When **a** drive is turned on, associates the Drive Selection (DK) and **Step Phase {PHASE) with the** contents of LTHCPY.

PORT UH = READ and STROBE SHIFT Register

This port will read the data from the floppy disk. You are pushed in **bit by bit** from the left.

PORT 12H = POLL DATA

via this port the synchronisation be1111 Read.

A negative pulse is generated when the next bit is available on port 11.

PORT 1JH = READ/WRITE PROTECT STATUS

This port allows you to write the write protection status of a floppy disk er1111 (write protection notch glued or not).

The result is passed in bit 7.
(1 = read-only)

Call and Overlooked

As already expected, the DOS occupies the address space '!ill00H in 5FFFH. There are 1n RM building blocks which are installed in the sketch control. The **1** sket control is connected to the system bus of the computer,

The process of rotating a floppy control is determined by checking a specific sequence {AA 55 E7 18). If this sequence of bytes is found, the initialisation routine that follows d1e is called automatically, which includes a BASIC vector, so that the} d1e spPzillen [@S commands can be detected and executed,

D1es 1st the RESTART !ill - Vector at address 78ill3H in the BASIC communication area. If the BASIC calls this vector within the order analysis (at address lDSAH), the DOS first checks whether a special DOS command is attached. If no, the programme, m, will continue with the normal BASIC command execution. If a DOS command is detected, then 111eroen ehe Required!chen Execution Routes **1** DÖS called.

The DOS vector range at the end of the memory is also built up from the DOS' initialisation rotors and filled with initial values.

You can also verify that a floppy control is connected to the system bus by using the appropriate **4MH** rhythm within a hash **programme**.

The DOS itself consists of a number of 1n closed subroutines. A large part of it can also be called from machine programmes, so that indiv1duelle floppy and data handling1n9en progra11 can be programmed and executed there.

You could edit the existing file system of the DOS dam1 t, but also create entirely new structures and processing forms, such as the files with direct access that were previously similar to the 111, which are not supported by the DOS.

Dip single Rutins directly at their start addresses would be In the guessing possibilities, you would get the programmes to a 05-version fierm, since s1h shifts in part of these addresses when the DOS changes.

A more elegant solution is the use of a new jumper bar at the beginning of the DOS, which was created by the tr manufacturer for such a purpose and allows the call of the nullified subroutines.

The call is made in m1t Help of the Z@-efehls

CAl.l. xxxxH.

The following subroutines can be reached via this jump bar:

Name	call	Function
RN	CALL 4008H	Power on drive
PROFF	CALL 400BH	Disable drive
ERROR	CALL 40EH	DOS Error Handling
RIAP	CALL 4011H	sector allocation grid load
CLEAR	CALL 4014H	Delete sector
SVMAP	CALL 4017H	sector allocation grid write
INIT	CALL 401AH	Initialise Diskette
CSI	CALL 401DH	Command Parameters interpret
HEX	CALL 4020H	ASCII conversion in HEX
IDAM	CALL 4823H	Find Adremar ke on Diskette
CREATE	CALL 402H	Write entry to table of contents
MAP	CALL 4029H	One free sector
SEARCH	CALL 402CH	File search in table of contents

FIND	CALL 402FH	Find *free* space in the content directory
RITE	CALL 4032H	Write sector to disk
READ	CALL 4035H	Read sector from disk
DLV	CALL 4038H	n Milliseconds Delay
STPIN	CALL 403RH	Move Head to Tracks
STPOUT	CALL 403EH	Reset Head to Tracks
LOAD	CALL 4M41H	to load a programme
SAVE	CALL 4044H	Save a programme!III

However, before calling any of these subroutines, very specific On- 9an9spar parameters must often be set, e.g. file name entry in DOS vector, drive selection in DK field, etc.,

It is also important to know what results such a subroutine will deliver where and what possible misperceptions may be reported.

You should also know which tabs are modified in the subroutines so that you can back them up first.

The following pages describe each of these subroutines with their input and output sizes, used registers, and error codes. This includes *a* function description and an application/call example.

Two additional functions, which you can not reach via the jump bar, but can easily progra1111111, are also listed. These are the runners ksausah! and checking the floppy disk write protection.

Note that **since$** does not change the IY tab in your programme. There, the initial address *of the* DOS vectors is entered at the initialisation, which is not only the DOS but also you when using *the* above. Routines constantly *need.*

From emn1gen routines, error codes are returned in register A. You should check this in any case whether your call was successful or not.

All data moving between the computer and the floppy disk uses the data buffer in the DOS workspace as a cache. Remember that every time you read or write a sector,$ changes its content.

The operating system generates an interrupt every 20 ms, which is normally used to update the screen content and evaluate the keyboard.

These interruptions, however, are with floppy access, where it depends on very precise time intervals, not 111ünschenS111ertj they would make error-free access impossible.

For this reason, you must disable interrupts and then enable them with enable interrupts (EI) before accessing any floppy disk.

In many cases, you must also check for yourself whether the disk to be edited is read-only; otherwise, write operations are performed anyway.

PRON

Drive Power

Call: **CALL 4088**

Input: DK IY+11) in DOS vectors= Drive Id **' 1** = Drive 1
Y' 80' = Drive 2

From5gang'

registers used: A

Troubleshooting: none

The scroll bar selected in OK will be enabled. The Drive1110tor is running and the red LED on the front of the drive is lit.

You should wait for a flight on this drive in time from a few Wed 111 seconds to Stabi l i of the rotation speed gkei t.

Example'

```
DI                      ;Disable Interrupts
LD      A, 80H          ;Select drive 2
LD      (IY+lll,A
CALL    4008H           Turn on {drive
LD      BC.50           Wait 5 ms
CALL    4038H
```

Drive 2 is turned on, then the programme *waits* 50 ms for stabilisation.

PWROFF

Turning off **a** Drive

Call: **r.ALl. 4111H**

Input:

Out:

Registers used: A Error

Handling: none

A powered drive is turned off. The drive motor stops and the LED on the front of the drive goes out.

Example:

```
CALL    4008RH          {Disable Drive
```

ERROR

DOS Error Handling

Call: or

CALL 4IIIIEH

JP 4IIIIIIEH

Input: A = Error Code (0 - 17)

Out: Jump 1n to BASIC interpreter Used

registers: AF, RC, DE, HL

Troubleshooting: none

According **to the** error code passed in the A-Register, an error message is received, A drive that may be powered on is turned off (if error code 1).

This routine differs from the other routines in that after the conclusion, the calling programme is not jumped to", but rather into the BASIC interpreter.
The stack pointer is set to the BASIC stack.

A=	generated message	Continue working
0	none	to the BASIC interpreter
	?SYNTAX ERROR	(!B9AH)
2	?FILE ALREADY	Issue and further sequence over 1997}
	EXISTS ?DIRECTORY	to the BASIC interpreter (1B9AH.l
3	FULL	- .
4	7DISK RITE	-
5	PROTECTED ?FILE NOT	- .
6	OPEN	-
7	?DISK IiO ERROR	
	?DISK FULL	- .
		-
		- .
		-

8	?FILE ALREADY OPEN
9	?DISK 1/0 ERROR
10	7DISK I/0 ERROR
11	7UNSUPPORTED DEVICE
12	7FILE TYPE MISMATCH
13	7FILE NOT FOUND
14	?DISK BUFFER FULL
15	?ILLEGAL READ
1	'ILLEGAL WRITE
17	BREAK on the BREAK routine 11DA0H) in BASIC

Example'

```
LD      A,7             ;Load Error Code 7
CALL    400EH           {tel: DISK FULL.
```

The l'lode '?DISK FULL' is issued and then BASIC reports to READY,

Note:

The line number field of the 742H in *the* BASIC communication area distinguishes from *the* ERROR routine, whether it is a direct command or a programme command.

If the field 7842H/7843H contains a value not equal to X'FFFF', this is interpreted as line zero!r and it is output behind **the** error message 1it (error codes 1 - 1) .

This FunHion can easily be used when testing out machine programmes by setting certain values in 7BA2H/78A3H, which will give you a Hrn!ile1s on the corresponding part of the programme if an error occurs.

Example'

```
        OR      A               ;Check for errors
        JR      Z,XY            ;no, continue'
        LD      HL, 10          ;Set Row ID
        LD      (78A2H),HL
        JP      4%EH            Call Error
XY
```

If A contains a value **that is** not equal, the appropriate error message will be displayed indicating the location of the occurrence.
usually A= 3, then "7DIRECTORY FULL IN 10'.

RDMAP

Sectors - Load occupation grid

Call: **CALL 4111H**

Input: The corresponding LauflilE!rk must be enabled.

Out: The sector occupancy grid is located in the 80-byte buffer at the end of the DOS workspace (MAPAREA1.

The drive remains on.

Registers used: AF, BC, DE, H

Error Handling< In the event of a failure, this routine will be used independently in
the ERROR routine branches. A separate evaluation is not possible.

The Sectors - Occupancy Chart (Occupancy Grid) is loaded from the sector 15 of the track 0 from the floppy disk into the memory and transferred to the MAPAREA at the end of the DOS workspace.

Note that you will need to power on and off the drive itself.

Example'

```
DI                      Undo Interruptly
LD      A, 1WH          Select Drive 1
LD      (IY+lll,A
CALL    4008H           and turn on
LD      C,50
CALL    4838H           Wait {50 ms
CALL    4011H           allocation grid   load
CALL    408BH           {Disable Drive
LD      L, ( IY+521     Start address Posting
LD      H,(IV+531       jaster in HL
EI                      ;Reset Interrupts 111
```

The Sector Occupancy Map 111is loaded from the floppy disk in drive 1, The initial address of the HAPAREA is then **provided** in the register pair HL.

Internally accessed routines: READ

CLEAR

Delete a Sector on the Floppy

Call: **CALL 411AM**

Input: The appropriate drive **must** be on.

SCTR IIY+17) = Sector accident
TRCK (IY+18) = Track number

Out: *The* addressed sector is physically deleted on the floppy disk.

Registers used: AF, Be, DE, HL

Troubleshooting: In the event of failure the routine **becomes** independent branched into ERROR routine. No error handling is possible.

The sector 111addressed in ***the*** DOS vectors SCTR IIY+17) and TRCK (IY+18) is physically deleted on ***the*** disk, i.e. overwritten with binary zeros **IX'00').**

Note that you need to power on and off the drive yourself.

Before deleting **the** sector, verify that the floppy disk is not read-only.

Example'

```
DI                      Disable Interrupts
LD      A,80H           ;Select drive 2
LD      ( IY+11.A
CALL    4008H           ;and power on
LD      BC.50           Wait 5 ms
CALL    4038H
IN      A, (13H)        ;Check write protection
OR      A
LD      A,4
JP      N,400EH         Read-only!
LD      (IY+17),12      Set {sec. to No
LD      (1Y+18),28      {Set Spurnuser
CALL    4014H           ;Delete sector
CALL    408BH           ;Turn off run111erk
EI                              {Turn back on interrupts
```

Sector 12 in track 28 of floppy disk 1n drive 2 is deleted.

Internally accessed routines: WRITE

SVMAP

Sectors - Write Usage Grid to Disk

Call: **CAl.l. 41117H**

Input: The appropriate drive must be on,

The current allocation grid is in the DOS workspace in NAPAREA.

Out: The sector occupancy grid was written in sector 15 of the track @ on the diskette i111 selected drive.

The scroll!rk remains on.

Registers used: AF, BC, DE, HL

Troubleshooting: In the event of an error, the routine branches directly into the ERROR routine. No error handling is possible.

The Sectors - Occupancy Chart (Occupancy Grid) is transferred from the corresponding DOS workspace (NAPAREAl) to the data buffer and then written to the sector 15 of the track on the disk.

Note that **since** you have to power on and off **the** drive itself 111must.

Before writing back, verify the floppy is not read-only.

Example:

```
DI                      ;Disable Interrupts
LD      A, 10H          Select {Drive 1
LD      (IY+ll),A
CALL    4008H           ;and power on
LD      BC.50           ;Wait 50 ms
CALL    4038H
IN      A, (13H)        ;Write         check
OR      A
LD      4
JP      M, 42EH         {read-only'
CALL    4017H           {Reverse Occupancy Grid
CALL    400BH           Disable {Drive
EI                      Turn on {interruts again
```

Sectors - Occupancy 111is written back from ,;:, DOSAubrication {MAPARE) to the floppy disk at Lau'eri (Trail , Sector 15).

Internally accessed routines: WR!TE

INIT

Initialise Diskette

Call: **CALL 411AH**

Input: DK (IY+ll) = Drive Id X'10' = Drive
X' 80' = Drive 2

Out: Initialised the floppy in the selected 111Urde drive.

Registers used: AF, RC, DE, HL, BC', DE', HL'

Troubleshooting: In the Fetile>'fall, the routine is corrected to the ERROR routine. A separate error handling Is not possible

A disk contained in the selected drive will be newly ini halized, i.e. it will be divided nrd 1n 4 tracks a 16 sectors and marked with the corresponding synrhronisat 1ons and markers.

All data previously on this chain will be deleted during this process.

This routine performs even power cycling the drive.

The scr, write protection and checked by INIT, the interruptions, *Pr dEn* Z initially switched off.

Example:

```
LD      A, 10H          Select Run Work
LD      ( IY ... 11
CAL     i,A 401AH       Reactivate {D1chain intial1Disable
L                       Interrupts
```

The floppy in drive 1 is initialised.

Internal routines: IDAM
STPIN
STPO\/
T DLY

Befer, interpret lparameter

Call: **CALL ltl1Dtl**

Input: HL = the initial address of a file name enclosed in a
single quotation mark.
This must be completed with X'00' or':'.

Out: The file name has been checked and transferred to
the **FNAl'l** field of the DOS vectors.

HL = Terminator Address

Registers used: AF, B, H

Troubleshooting: If the data name is not enclosed in quotation marks or is not correctly completed, then 111is branched to BASIC and the message "?SYNTAX ERROR" is returned.

The first eight characters of a file name enclosed in quotation marks are transferred to the **FNA}** field of the DOS vectors.

Behind **dell!** final quotation marks must be a BASIC end - of - line identifier **X'00** or a command separator ':'.

This routine is used by the DOS-BASI for synapse testing.

Floppy access does not occur.

Example

```
          LD     HL,DNM1          {File Name Address 1n
          CALL 401DH              FNM Transfer


DNAl11    DEFM ''NAILBOX'':
```

The filename 'MAILBOX' is transferred to the DOS vector FNAM for subsequent addressing.

HEX

ASCII to HEX Call: **CALL**

4828H

Input: HL = Initial address of a 4 byte hexadecimal number in ASCII format.

Out: DE = equivalent hexadecimal value (binary) HL = Input address+ 4

Registers used: AF, DE, HL

Troubleshooting: Carry = - no error

Carry = 1 - Conversion Error

This routine can be used to convert a hexadecimal input from the keyboard to the binary equivalent.

By DOS-BASIC, **this** routine is used, for example, by BSAVE command to interpret and adopt the programme's initial address and end address.

Example:

```
        LD      HL, ASCII       address iASCII
        CALL    4020#           i wr,wande 1 n
        JR      NC, A1          {Carry@? Okay, to A!
        LD      A+ 1            ;Carry=l, "SYNTAX ERROR'
        JP      400EH           ;Output via ERROR routine
A1      LD      (BIN),DE        Save Binary Value
ASCII   DEFI'I  'A31C'
BIN     DEFS    2
```

The string in the ASCII field is converted to the hexadecimal value and stored in the BIN field.

IDAM

find the e}mark on the disk

Call: CALL 4623%

Input: The appropriate drive must be on,

SCTR (IV+17) = sector number
TRCK (IY+18) = trace number

Out: in case of error-free111 return jump, the read/write head is located directly in front of the data mark of the desired sector.

Registers used: AF, BC, DE, H

Troubleshooting:

A = 11A = 9	three-mark found Adjust value not found BREAK key
A = 17 -	pressed

if A = 0, Z flag is set if A {} ,
Z flag is deleted

This routine is used to position u11 before **defi** write or read a sector the write/read header before *the* data mark of that sector.

IDA first positions the head above the desired track and reads then address mark by address mark until the desired sector has been found, then the write or read process for the data field must start immediately, **since the** floppy disk is spinning.

IDAM is already integrated into the READ and WRITE routines for reading and writing a sector.

Example:

```
DI                      ;Disable Interrupts
LD      A, 80H          ;Select drive 2
LD      (lYtll l,A
CALL    4008H           Turn on {drive
LD      BC.50           ;Wait 50 ms
CALL    4038H
LD      IY+17),         {Address sector
LD      (IY+18),14      {Address Track
CALL    4023H           Find sector
JP      NZ, 400EH       ;Error or BREAK

        Read or Write Sector


The write/read opt should read to the attached
or write before the data mark of the sextor 6 of track 14.


Internal routines:              STPIN
                                STPOUT
```

CREATE

Write entry to table of contents

Call: **CALL 4826H**

Input: File name 1n FNAM (IY+l) File type in TYPE (IY+9)

The 1Uj occupancy grid should be loaded

(MAPAREAi.

The drive **must** be on.

Out: The **file has been** entered **in** the table of contents, the first data sector for **this file has been** reserved.

Registers used: *Pf*, BC, DE, Hl.

Troubleshooting: = 0 - entry carried out
= 2 - File already present
A= 3 - No 1!1! Table of Contents
A = 7 - no sector free (floppy full) A=9 - An address mark has been found
A = 1 - a checksum error occurred on A = 17- the BREAK key was pressed

For the file 1rd specified in FMAM, give an entry to the table of contentsOll'll and the first free sector reserved for this file.

First, 1Dit SEARCH checks whether a file with the same name already exists. If not, FIND identifies a free entry 1m Table of Contents and 111i t 11AP searches for a first free sector. If both were successful, the entry in the Int,altsverzeichnis 11orgen Omen,

To do this, enter the data type, separator ':', file name and the address of the first sector in the unmarked lb-byte entry of the table of contents, and the table of contents 111is written back.

The occupancy grid should then also be written back to the floppy disk, otherwise the first sector does not definitively occupied 1st, forget this, so it will later lead to a double occupancy of this sector and da111it may lead to a non-ent111erroneous mess of the data.

Slot ears can of course take advantage of this and make two different entries for the same file with different names in the table of contents. meaningful 73777

Before calling CREATE, you should definitely check the write protection of the chain.

From111 evaluation of the A-Register can be used to verify the success of the action.

```
          Tue                       ;Turn Off Interrupts
          LD     (JY+l!J, !0H       Select {Drive 1 and
          CALL   4008H              Turn On
          LD     R, 50              ;50 ms pause
          CALL   4038H
          IN     A, (13HI           Check {Write Protection
          OR     A
          LD     A,
          JP     , 400EH            ;read-only ' ;Load
          CALL   4011H              allocation grid {file
          LD     HL,DNAM            name in field ;Apply
          CALL   401DH              FNAM
          LD     (IY+9),'R          ;Type= Set 'B'
          CALL   4026H              Add iFile to Table of Contents ;Failed?
          OR     A                  ija, on the ERROR routine
          JP     N? , 400EH         ;Revert Usage Grid ;Disable Drive
          CALL   4017H              ;Allow Interrupts 111
          CALL   400BH
          EI
DNAM      DEFM   "CARD":'
```

For the binary file 'KARTEI' (type= ß) an entry in the table of contents of the disk 1m drive 1 is made.

Internal routines:

SEARCH
FIND
MAP
READ
WRITE

MAP

Identify a frBen sector on the floppy disk

Call: `CALL 4029`

Input: The assignment grid must be in the DOS workspace (MAPAREAI).

Out: NSCT (IY+2J) sector number

NTRK {IY+!Z" = Track Number

- The sector addressed to NSCT IIII t uno NTRK has been reserved in the internal allocation grid (iMAPAREA).

Registers used: AF, RC, H

Error bet,Action:

A = 0	Found Sector
A= 7	no sector left free

This routine contains a free sector 1n of the internal relegation overs1cht of the BÖS (MAPAREA), which should previously be filled with the current occupancy grid of track , sector 15 of the disk.

If a free sector is determined, the corresponding bit in the occupancy summary is set to 1.

Note, however, that a final occupancy has not been made until the occupancy grid is written back to the floppy disk ur de.

The result (the sector address) is passed in the *NSCT* and NTRK fields of the DOS vectors. If you want to access the sector, e.g. with WRITE, you have to transfer this address in the fields SCTR and TRECK beforehand.

The MAP 111 routine does not perform floppy access.

Example

```
DI                      ;Disable Interrupts
LD      (1Y+11),80H     {Drive? select
CALL    4008H           ;and power on
LD      BC.50
CALL    4038H           {50 ms pause
CALL    4011H           ;Load allocation grid
CALL    4029H           Determine Which Sector Is Free
OR      A               ;Error?
JP      NZ, 400EH       ;Yes, to ERROR routine
CALL    4017H           ;Reverse Occupancy Grid
CALL    480RH           ;Disable drive
EI                      Turn on {Interrupts}
```

On the disk in drive 2 a free sector is identified and occupied. The address of the sector is passed in the NSCT and NTRK fields of the DOS vectors.

SEARCH

Find File in Table of Contents

Call: **CALL 41201**

Input: The drive **must** be on.

FNM (IY+1) = File Name

Output if file exists, type of file in TYPE+! IIY+101.

The sector of the table of contents with delll found entry is i • Data buffer.
DE points to the byte behind the man.
SCTR and TRCK contain the address of the sector.

Registers used: A, B, DE, ll.

Troubleshooting: A= - File not present
A = 2 - File present
A =9 - Address Mark not found
A = 10 - Checksum111111en error reading A
= 13 - File not present
A = 17 - The BREAK key is pressed

The SEARCH routine checks if the i111 table of contents of the addressed disk already exists a file with de • stored in FNAN.

The result of the search is passed in the A-Register.

A = 2 means that an appropriate entry exists.
The sector of the table of contents is in the data buffer and DE points to the byte behind the name of the entry found (= address of 1. sector of this file).
The SCTR and TRK fields of the DOS vectors contain the sector address within the table of contents.
The TVPE+I (IY+1) field contains the type of file found.
You may have to evaluate it yourself.

A = 8 or A = 13
They have the same meaning. The specified file is not included in the contents of the disk. A=0 - reaches the end of valid entries 1111.1rde. A=13 - the end of the table of contents was
.

All other values of A 1!le1sen indicate an error or press the BREAK button.

Example

```
DI                      Disable {Interrupt
LD      IY+11, 10#      Select Drive 1
CALL    888H            ;and power on
LD      BC, 5
CALL    4038H           {50 Ms Pause
CALL    4011H           ;allocation grid  load
```

```
        LD      HL,DNAM         Transfer filename to FNA
        CALL    401DH
        CALL    4026H           Find {Date1 in the Contents Directory
        OR      A
        JR      Z,A1            ;not existing'
        CP      0DH
        JR      Z,A1            ;not present'
        CP      2               Feh]er?
        JP      NZ, 400EH       ;yes, to ERROR routine
        IN      A, (13H)        ;Check write protection
        OR      A
        LD
        JP      t, 400EH        ;read-only, to ERROR routine
        EX      DE,HL           ;Address of the entry in HL
        LD      EN,-10          ;HL to top of entry
        ADD     HL, DE
        LD      (HL),1          Share Entry
        CALL    4032H           {Sector of Contents          back.

                occupied sectors Release 1m allocation grid

A1      CALL    4017H           {Reverse Occupancy Grid
        CALL    400.BH          ;Disable drive

        EI                          {Turn back on interrupts

DA      DEFN    '·DIARY' '
```

If present, the file 'DIARY' will be deleted from the 1m table of contents of the floppy disk in the 111erk 1. If it does not exist, the delete routine is skipped.

Note that this example has not been fully encoded. In addition to deleting the entry in the table of contents,
You also have to release all occupied sectors of this file into allocation grid.

Internal routines: READ

FIND

find a free entry in the table of contents

Call: **CALL 412FH**

Input: The drive must be on

Out: SCTR = sector number TRCK = trace number

The addressed sector of the table of contents is in the data buffer,
Hl points to the beginning of the free entry.

Registers used: AF, C, DE,

Troubleshooting:		
	A = A = 3 = 9	ok, free entry does not find free entry
	A= 10 A = 17 -	an address mark was not found read Checksum error occurred BREAK button pressed

A free entry is found in the table of contents of the diskette that is powered on.

1111 Register A, the result shall be transmitted.

If successful (A = 0), the corresponding sector of the table of contents is in the data buffer, and register pair HL points to the beginning of the free entry.

The fields SCTR and TRCK contain the address of this sector. You can now add an entry there.

Example'

```
DI                      ; Disable Interrupts
LD     (IY+lli,10H      Select Lauterk 1
CALL   4008H            and turn on
LD     RC, 50
CALL   4038H            ;50 ms pause
CALL   402FH            find ;free entry
OR     A                Successful?
JP     NZ,400EH         no" to ERROR routine
CALL   400BH            ;Disable drive
EI                      Reset Interrupts
```

A free entry in the table of contents is found on the disk in drive I.

Internal routines: READ

WRITE

Write sector to disk

Call: l'.Al.L. 4132H

Input: The volume must be turned on.

The sector to be written must be in the data buffer.

The address of the sector to write must be in the SCTR and TRCK fields of the DOS vectors.

Out: The sector in the data buffer has been transferred to the disk.

Registers used: AF, BC DE, HL, BC', DE', HL'

Troubleshooting:

A = 0 ok, sector written
A = 9 Adrejlmark not found
A = 17 - BREAK 1111.1rde confirmed

The data in data buffer 1>etindl1chen 1t1are transferred to the addressed sector
of the floppy disk. The checksum is determined and submitted to the sector end.

The sector 111 is transferred including the data mark (1 bytes) and the checksum before the data buffer, i.e. a total of 14 bytes
who wrote the disk to the disk.

Example?

```
DI                      ;Disable Interrupts
LD      ( IY+11 i, 10H  Select Drive 1
CALL    4008H           ;and power on
LD      BC, 50
CALL    4038H           ;50 ms pause
IN      A, (13H)        ;Write         check
OR      A
LD      A,4
JP      #4@0EH          Read-only!
LD      (IY+17),10      {Sector number in SCTR
LD      (IY+18),5       ;track number in TRCK
CALL    4032H           {Write sector to disk
0R      A               Feh]er occurred?
JP      NZ,400EH        yes, to the ERROR routine
CALL    400BH           Disable drive
EI                      {Turn back on interrupts
```

The data from the data buffer is written to track 5_1 sector 10 of the disk in drive 1.

Internal routines: IDAM

READ

Read sector from disk

Call: **CALI 4135H**

Input: The drive should be on.

Address of the sector to be read in SCTR (IY+17) and
TRCK !IY+18) of DOS vectors.
RETRY (IY+19) = Number of read attempts

Out: The read sector is in the data buffer.

Registers used: AF, BC, DE, HL

Troubleshooting:		
	A = @A = 9	ok, sector not found read address tag checksum wrong
	A = 1 A = 17	the BREAK key **has been** pressed

The 11it SCTR and TRCK addressed sector is transferred from the disk to the data buffer.

The test tube is obtained and the value stored there is compared at the end of the sector.

111 field RETRY (IY+191 of DOS vectors), you can specify how many read attempts should be performed in the event of a wrong test (default value = 10).

Example

```
DI                      ;Turn off interrupts
LD      (IY+11),808     Select {drive 2 ;and
CALL    4008H           power on
LD BC,50                450 ms Pause
CALL    4838H           iSectoral painter in
LD      (IY+17),14      SCTR
LD      (lY+l,28        { Track number in TRK
```

```
CALL    4035H           Read the sector
OR      A               ;Failed?
JP      NZ, 400EH       yes, to turn off ERROR routine
CALL    488BH           drive turn interrupts back on
EI
```

The data of sector 14 on *the* track 28 is transferred from *the* disk in drive 2 to the data buffer.

Internal routines: IDAll

DLV

n Milliseconds Delay

Call: **CALL** 4838

Input: BC = Number of milliseconds

usgan9'

Registers used: *AF*, RC Error

Handling:

This routine causes a delay, the duration of which is determined in milliseconds by the entry in the register pair BC.

Example:

```
DI                      ; Disable Interrupts
LD      BC, 1000
CALL    4038H           ;1 Wait {Allow Interrupts
EI
```

causes *a* delay of one second.

STPIN

Put the write/ read header in the tracks

Call: **CALL 4131H**

Input: The drive must be powered on s1n.

BC = Number of tracks

Out: The write/read head has been set the appropriate number of tracks.
DTRCK (IY+20) contains the new current track number.

Registers used: AF, BC

Troubleshooting:

The write/read header **is** set around the number of tracks contained in mn B, but not more than 39.

Example:

```
DI                      Disable {Interrupt
LD      IY+11), 10%     {Drive 1 Select
CALL    4888H           and turn on
LD      BC, 58
CALL    4038H           {50 Ms Pause
LD      B,1             ;Put head 10 tracks
CALL    403BH
CALL    400BH           {Disable Drive
EI                      Allow Interrupts
```

STPOUT

Write/ Read Head Reset in Tracks

Call: **CALL 48JEH**

Input: The drive must be on.

BC = Number of tracks

Out: The write/read opt reset the corresponding number of tracks.
DTRCK (IY+20} contains the new current track number.

Registers used: AF, BC

Troubleshooting:

The write/read header is reset by the number of tracks in B, but not more than 0.

Example

```
DI                      Disable {Interrupt
LD      CIY+lll,10H     Select Drive 1
CALL    4008H           ;and power on
LD      BC, 50
CALL    4038H           i50 ms break
LD      1.5             Reset Header 5 5
CALL    403EH
CALL    400BH           ;Disable drive
EI                      ;Allow Interrupts
```

LOAD

Loading a programme *or* memory area

Call: **CALL 4141H**

Input: File name in FNAII (IY+1) File type in TYPE (IY+)

Out: 78A4/A5 = initial address 78F9/FA =end address+ 1

Registers used: AF, RC, DE, HL

Troubleshooting: **A = 8** o
k

A =9 Address label not found
A = 10 - PrütsuMen Error
A= 12 - File found, but incorrect type A = 13
- File not present
A = 17 - BREAK button pressed

The **programme** identified in FNN field is transferred from the disk to memory.

After successful execution, the start address will be passed to address 78A4/A5 in the BASIC communication area and the end address+1 of the memory area will be passed to address 78F9/FA.

This routine only works for those files that have a table of contents in bytes 12 and 13 the initial address and in bytes 14 and 15 contain the end address+1 *of the* file, i.e. not for standard data files.

If a programme is saved with the BASIC commands SAVE or RSAVE or by machine programmes in the routine SAVE, this entry is automatically saved.

The Rutine LOA automatically turns the drive on and off and also turns off the interruptions.

Example;

```
LD      (IY+111,80H       {Drive? select
LD      HL,DNAl'1         ;File name in FNAN
CALL    401DH            to enter ;s
LD      <IY+91,'B'        ;Type= 'B' (binary file 1
CALL    4041H             Load { programme
OR      A                 Error occurred 2
JP      NZ, 400EH         yes, to the ERROR routine
EI                        Reset Interrupt
```

DNAN ''GRAFDR'':

The binary file or the GRAFDR' machine programme is moved from *the* floppy disk to drive 1.

Internal routines: SEARCH
READ

SAVE

Saving a Progranas of the Storage Area to Diskette

Call: **CALL 404**

Input: 78AA/A5 = Initial address of 78F9/FA memory range = End address+i of memory range

The drive **must** be on.

FNAM (IY+1) = file/programme name
TYPE <IY+91 = File type

Out: The addressed memory area was transferred to the floppy disk and entered under **the** specified name in the table of contents.

Registers used: A, BC, DE, HI.

Troubleshooting: If errors occur 111the ERROR routine is branched directly from the SAVE Rouhne. A separate error handling 1st makes possible.

A memory area or programme is transferred from memory to the floppy disk. Start and end addresses are passed in the corresponding BASIC indicators 78A4 and 78F9.

This area is entered in the table of contents under the NaMen specified in FNAM and the **1** first bytes of TYPE.

You must power on the floppy disk 111 yourself; it is turned off by the SAVE routine when the save process is complete.

You should also check the write protection before doing so.

Be1spiel'

```
        LD      H, 000H          :Start Address
        LD      (7844#),HL
        LD      HL,90001-!       ; End Address+ 1
        LD      (78F91, HL)
        LD      HL, DNAM         {File Name in FNA
        CALL    401DH            ; transmit
        LD      (1Y+9),'R        ;Type= Binary
        LD      (IY+11),1MH      ;Select Drive 1
        CALL    4008H            and a nscha 1 ten
        IN      A,(13Hi          ;Check write protection
        0R      A
        LD      A,4
        JP      #, 400EH         read-only'
        CALL    4044#            ;Writes memory to disk
        EI                       Turn on {Interrupts}
DNAM    DEFM    'TESTDA T':'
```

The memory range 8000 - 8FFF **is** transferred under **de!l** name 'TESTDAT' with det11 type 'B' skin.

Internally accessed routines:

READ	CREATE
MAP	SEARCH
wRITE	

To complement these routines, here are two more functions that you can find in many of the previous examples.

DRIVE

Select a Drive

This function is not accessible via the jump bar.

However, it is easy to accomplish, because only the correct code of the runtime111erks is to be entered in the DK (IY+II) field of the DOS vectors.

```
Drive = LD (IY+11),10H drive? = LD

(IY+11),8MH
```

Using this code, the PWRON routine selects the correct drive and turns it on.

WPROCT

Check write protection

You are often responsible for checking the write-protection status of a floppy disk before performing a write operation.

The information about this can be obtained via port 13. If the write-protect notch of the floppy disk is superglued, then bit 7 of this port passes a 1.

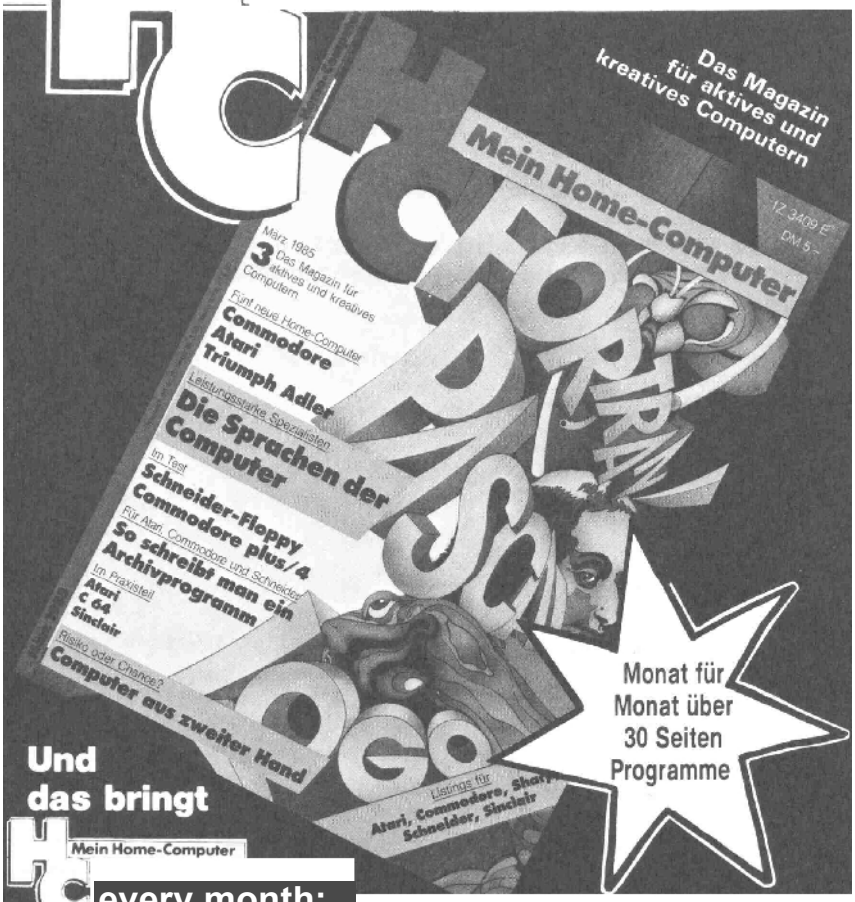
The E!rk aui will be selected and enabled.

Example

```
IN      A, (13H)        ;Scan port 13
OR      A               {Byte ;Set error
LD      A,4             code
JP      #, 400EH        if negative, then d1e ;disk
                        is read-only.
```

If the disk is read-only, error code 4 **is** branching into the ERROR routine and the message 'DISK RITE PROTECTED' is printed there.

No Home Computers

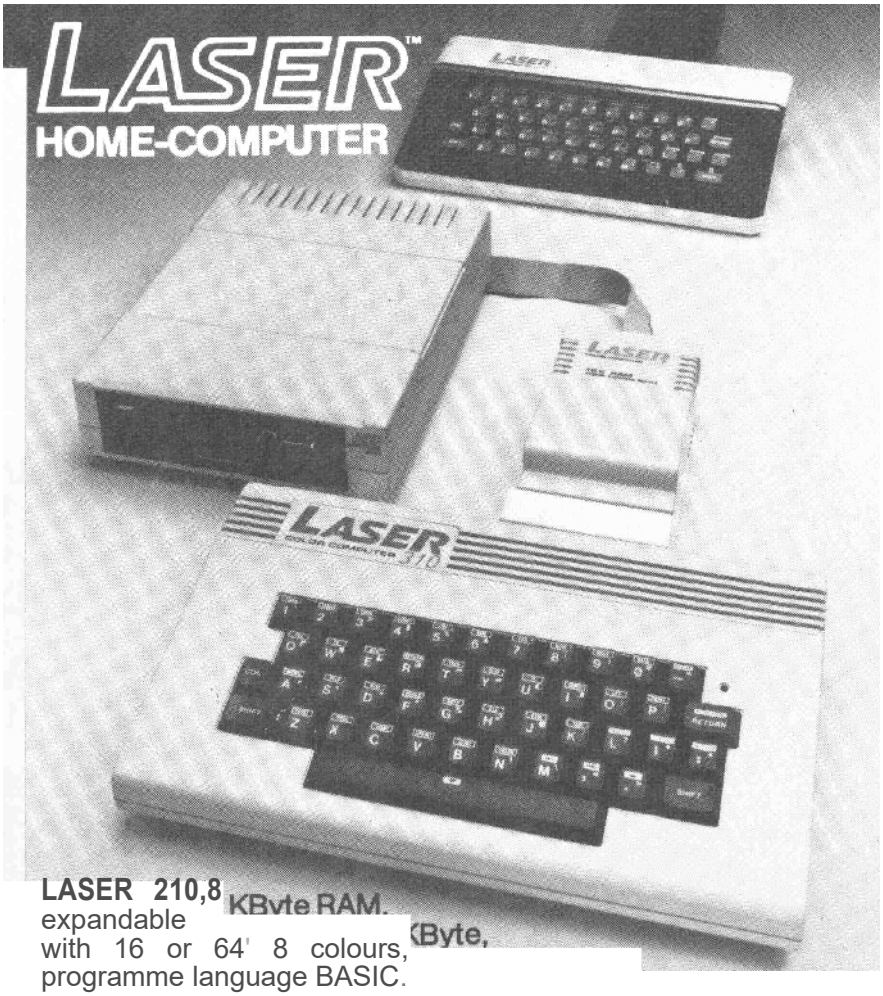


every month:

- Programmes for all common home computers * Practical examples
- Market overview, tests and purchase advice for ancillary equipment and home computers
- Getting Started Fast Courses * Tips and Tricks
- Interesting, current and entertaining things from the home computer scene
- News, Club News

Get the latest issue from your magazine dealer or request a brochure directly from VogelVerlag, Leserservice HC, Postfach 67 40, 8700 Würzburg.

The entry-level models for students



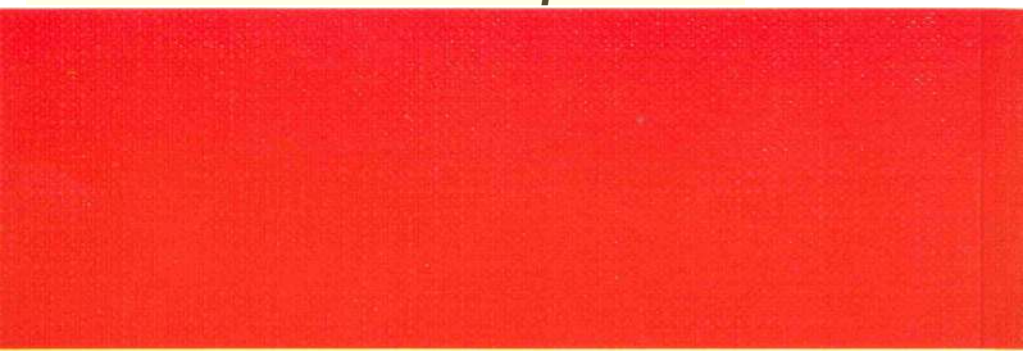
LASER 210,8 KByte RAM. expandable KByte, with 16 or 64 8 colours, programme language BASIC.

LASER 310 with the same equipment as Laser 210, but 18 KByte RAM and with typewriter keyboard.

2 Drive **Floppy Disk Controller** with LASER-DOS, memory capacity 80 Kbytes.

Importer-General: CE TEC Trading GmbH
Lange Reihe 29, 2000 Hamburg 1

active and creative computers



Sooner or later, every computer owner's desire for a floppy disk storage becomes overpowering. The advantages of direct access over sequential operation are too great. This is the only way to take full advantage of all the possibilities of your device.

In this volume, the diskette operating system of the laser computer is described in detail in its structure and application. In addition to the BASIC-DOS-commands, the interface and applicability in machine programmes are also explained. Use examples make it easier to enter the floppy world.

(@g@ **VOGEL-BUCHVERLAG W\JRZBURG**

ISBN 3-8023-0868-9